

Network Working Group
INTERNET-DRAFT
Updates: 2865,3162,6158,6572
Category: Standards Track
<draft-dekok-radext-datatypes-06.txt>
1 April 2015

DeKok, Alan
FreeRADIUS

Data Types in the Remote Authentication
Dial-In User Service Protocol (RADIUS)
draft-dekok-radext-datatypes-06.txt

Abstract

RADIUS specifications have used data types for two decades without defining them as managed entities. During this time, RADIUS implementations have named the data types, and have used them in attribute definitions. This document updates the specifications to better follow established practice. We do this by naming the data types defined in RFC 6158, which have been used since at least RFC 2865. We provide an IANA registry for the data types, and update the RADIUS Attribute Type registry to include a "Data Type" field for each attribute. Finally, we recommend that authors of RADIUS specifications use these types in preference to existing practice.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on October 1, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info/>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Specification use of Data Types	4
1.2. Implementation use of Data Types	4
1.3. Requirements Language	5
2. Data Type Definitions	6
2.1. integer	7
2.2. enum	8
2.3. ipv4addr	8
2.4. time	9
2.5. text	10
2.6. string	10
2.7. concat	11
2.8. ifid	12
2.9. ipv6addr	13
2.10. ipv6prefix	14
2.11. ipv4prefix	15
2.12. integer64	16
2.13. tlv	16
2.14. vsa	18
2.15. extended	19
2.16. long-extended	20
2.17. evs	22
3. Updated Registries	24
3.1. Create a Data Type Registry	24
3.2. Updates to the Attribute Type Registry	25
4. Suggestions for Specifications	30
5. Security Considerations	31
6. IANA Considerations	31
7. References	31
7.1. Normative References	31
7.2. Informative References	32

1. Introduction

RADIUS specifications have historically defined attributes in terms of name, type value, and data type. Of these three pieces of information, only the type value is managed by IANA. There is no management of, or restriction on, the attribute name, as discussed in [RFC6929] Section 2.7.1. There is no management of data type name or definition. This document defines an IANA registry for data types, and updates the RADIUS Attribute Type registry to use those newly defined data types.

In this section, we review the use of data types in specifications and implementations. We highlight ambiguities and inconsistencies. The rest of this document is devoted to resolving those problems.

1.1. Specification use of Data Types

A number of data type names and definitions are given in [RFC2865] Section 5, at the bottom of page 25. These data types are named and clearly defined. However, this practice was not continued in later specifications.

Specifically, [RFC2865] defines attributes of data type "address" to carry IPv4 addresses. Despite this definition, [RFC3162] defines attributes of data type "Address" to carry IPv6 addresses. We suggest that the use of the word "address" to refer to disparate data types is problematic.

Other failures are that [RFC3162] does not give a data type name and definition for the data types IPv6 address, Interface-Id, or IPv6 prefix. [RFC2869] defines Event-Timestamp to carry a time, but does not re-use the "time" data type defined in [RFC2865]. Instead, it just repeats the "time" definition. [RFC6572] defines multiple attributes which carry IPv4 prefixes. However, an "IPv4 prefix" data type is not named, defined as a data type, or called out as an addition to RADIUS. Further, [RFC6572] does not follow the recommendations of [RFC6158], and does not explain why it fails to follow those recommendations.

These ambiguities and inconsistencies need to be resolved.

1.2. Implementation use of Data Types

RADIUS implementations often use "dictionaries" to map attribute names to type values, and to define data types for each attribute. The data types in the dictionaries are defined by each implementation, but correspond to the "ad hoc" data types used in the specifications.

In effect, implementations have seen the need for well-defined data types, and have created them. It is time for RADIUS specifications to follow this practice.

This document requires no changes to any RADIUS implementation, past, present, or future. It instead documents existing practice, in order to simplify the process of writing RADIUS specifications, to clarify the interpretation of RADIUS standards, and to improve the communication between specification authors and IANA.

1.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Data Type Definitions

This section defines the new data types. For each data type, it gives a definition, a name, a number, a length, and an encoding format. Where relevant, it describes subfields contained within the data type. These definitions have no impact on existing RADIUS implementations. There is no requirement that implementations use these names.

Where possible, the name of each data type has been taken from previous specifications. In some cases, a different name has been chosen. The change of name is sometimes required to avoid ambiguity (i.e. "address" versus "Address"). Otherwise, the new name has been chosen to be compatible with [RFC2865], or with use in common implementations. In some cases, new names are chosen to clarify the interpretation of the data type.

The numbers assigned herein for the data types have no meaning other than to permit them to be tracked by IANA. As RADIUS does not encode information about data types in a packet, the numbers assigned to a data type will never occur in a packet.

The encoding of each data type is taken from previous specifications. The fields are transmitted from left to right.

Where the data types have inter-dependencies, the simplest data type is given first, and dependent ones are given later.

We do not create specific data types for the "tagged" attributes, as discussed in [RFC2868]. That specification defines the "tagged" attributes as being backwards compatible with pre-existing data types. In addition, [RFC6158] Section 2.1 says that "tagged" attributes should not be used. There is therefore no benefit to defining additional data types for these attributes.

Similarly, we do not create data types for some attributes having complex structure, such as CHAP-Password, ARAP-Features, or Location-Capable. We need to strike a balance between correcting earlier mistakes, and making this document more complex. In some cases, it is better to treat complex attributes as being of type "string", even though they need to be interpreted by RADIUS implementations.

Implementations not supporting a particular data type MUST treat attributes of that data type as being of data type "string". See Section 2.6, below for a definition of the "string" data type.

The definitions below use specialized names for various fields of attributes and data types. These names serve to address ambiguity of

the field names in previous specifications. For example, the term "Value" is used in [RFC2865] Section 5 to define a field which carries the contents of attribute. It is then used in later sections as the sub-field of attribute contents. The result is that the field is defined as recursively containing itself. Similarly, "String" is used both as a data type, and as a sub-field of other data types.

This document uses slightly different terminology than previous specifications, in order to be avoid ambiguity. The first addition is the following term:

Attr-Data

The "Value" field of an Attribute as defined in [RFC2865] Section 5. The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry.

In this document, we use the term "Value" only to refer to the contents of a data type, where that data type cannot carry other data types. In other cases, we refer to the contents of a data type as "Type-Data", to distinguish it from data of other types. For example, a data type "vsa" will contain a data field called "VSA-Data".

These terms are used in preference to the term "String", which was used in multiple incompatible ways. It is RECOMMENDED that future specifications use the new terms in order to maintain consistent definitions, and to avoid ambiguities.

2.1. integer

The "integer" data type encodes a 32-bit unsigned integer in network byte order. Where the range of values for a particular attribute is limited to a sub-set of the values, specifications MUST define the valid range. Values outside of the allowed ranges SHOULD be treated as invalid.

Name

integer

Number

1

Length

Four octets

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Value                                                    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.2. enum

The "enum" data type encodes a 32-bit unsigned integer in network byte order. It differs from the "integer" data type only in that it is used to define enumerated types, such as Service-Type. Specifications MUST define a valid set of enumerated values, along with a unique name for each value.

Name

enum

Number

2

Length

Four octets

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Value                                                    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.3. ipv4addr

The "ipv4addr" data type encodes an IPv4 address in network byte order. Where the range of address for a particular attribute is limited to a sub-set of possible addresses, specifications MUST define the valid range(s). Values outside of the allowed range SHOULD be treated as invalid.

Name

ipv4addr

Number

3

Length

Four octets

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Address           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.4. time

The "time" data type encodes time as a 32-bit unsigned value in network byte order and in seconds since 00:00:00 UTC, January 1, 1970. We note that dates before the year 2013 are likely to be erroneous.

Note that the "time" attribute is defined to be unsigned, which means it is not subject to a signed integer overflow in the year 2038.

Name

time

Number

4

Length

Four octets

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Time           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.5. text

The "text" data type encodes UTF-8 text [RFC3629]. The maximum length of the text is given by the encapsulating attribute. Where the range of lengths for a particular attribute is limited to a sub-set of possible lengths, specifications MUST define the valid range(s).

Note that the "text" type does not terminate with a NUL octet (hex 00). The Attribute has a Length field and does not use a terminator. Texts of length zero (0) MUST NOT be sent; omit the entire attribute instead.

Name

text

Number

5

Length

One or more octets.

Format

```

0
0 1 2 3 4 5 6 7
+-----+
| Value   ...
+-----+
```

2.6. string

The "string" data type encodes binary data, as a sequence of undistinguished octets. Where the range of lengths for a particular attribute is limited to a sub-set of possible lengths, specifications MUST define the valid range(s).

Note that the "string" data type does not terminate with a NUL octet (hex 00). The Attribute has a Length field and does not use a terminator. Strings of length zero (0) MUST NOT be sent; omit the entire attribute instead.

Where there is a need to encapsulate complex data structures, and

TLVs cannot be used, the "string" data type MUST be used. This requirement include encapsulation of data structures defined outside of RADIUS, which are opaque to the RADIUS infrastructure. It also includes encapsulation of some data structures which are not opaque to RADIUS, such as the contents of CHAP-Password.

There is little reason to define a new RADIUS data type for only one attribute. However, where the complex data type cannot be represented as TLVs, and is expected to be used in many attributes, a new data type SHOULD be defined.

These requirements are stronger than [RFC6158], which makes the above encapsulation a "SHOULD". This document defines data types for use in RADIUS, so there are few reasons to avoid using them.

Name

string

Number

6

Length

One or more octets.

Format

```
0
0 1 2 3 4 5 6 7
+---+---+---+---+
| Octets    ...
+---+---+---+---+
```

2.7. concat

The "concat" data type permits the transport of more than 253 octets of data in a "standard space" [RFC6929] attribute. It is otherwise identical to the "string" data type.

If multiple attributes of this data type are contained in a packet, all attributes of the same type code MUST be in order and they MUST be consecutive attributes in the packet.

The amount of data transported in a "concat" data type can be no more than the RADIUS packet size. In practice, the requirement to

transport multiple attributes means that the limit may be substantially smaller than one RADIUS packet. As a rough guide, is RECOMMENDED that this data type transport no more than 2048 octets of data.

The "concat" data type MAY be used for "standard space" attributes. It MUST NOT be used for attributes in the "short extended space" or the "long extended space". It MUST NOT be used in any field or subfields of the following data types: "tlv", "vsa", "extended", "long-extended", or "evs".

Name

concat

Number

7

Length

One or more octets.

Format

```

0
0 1 2 3 4 5 6 7
+-----+
| Octets   ...
+-----+
```

2.8. ifid

The "ifid" data type encodes an Interface-Id as an 8-octet string in network byte order.

Name

ifid

Number

8

Length

Eight octets

Format

```

      0                               1                               2                               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      Interface-ID ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      ... Interface-ID
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.9. ipv6addr

The "ipv6addr" data type encodes an IPv6 address in network byte order. Where the range of address for a particular attribute is limited to a sub-set of possible addresses, specifications MUST define the valid range(s).

Name

ipv6addr

Number

9

Length

Sixteen octets

Format

```

      0                               1                               2                               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      Address ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      ... Address ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      ... Address ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      ... Address
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.10. ipv6prefix

The "ipv6prefix" data type encodes an IPv6 prefix, using both a prefix length and an IPv6 address in network byte order.

Name

ipv6prefix

Number

10

Length

At least two, and no more than eighteen octets.

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Reserved   | Prefix-Length | Prefix ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
... Prefix ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
... Prefix ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
... Prefix
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Subfields

Reserved

This field, which is reserved and MUST be present, is always set to zero.

Prefix-Length

The length of the prefix, in bits. At least 0 and no larger than 128.

Prefix

The Prefix field is up to 16 octets in length. Bits outside of the Prefix-Length, if included, must be zero.

2.11. ipv4prefix

The "ipv4prefix" data type encodes an IPv4 prefix, using both a prefix length and an IPv4 address in network byte order.

Name

ipv4prefix

Number

11

Length

At least two, and no more than eighteen octets.

Format

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Reserved   | Prefix-Len | Prefix ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
... Prefix      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Subfields

Reserved

This field, which is reserved and MUST be present, is always set to zero.

Prefix-Length

A 6-bit unsigned integer containing the length of the prefix, in bits. The values MUST be no larger than 32.

Prefix

The Prefix field is 4 octets in length. Bits outside of the Prefix-Length must be zero. Unlike the "ipv6prefix" data type, this field is fixed length. If the address is all zeros (i.e. "0.0.0.0", then the Prefix-Length MUST be set to 32.

2.12. integer64

The "integer64" data type encodes a 64-bit unsigned integer in network byte order. Where the range of values for a particular attribute is limited to a sub-set of the values, specifications **MUST** define the valid range(s).

Name

integer64

Number

12

Length

Eight octets

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Value ...                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     ... Value      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

2.13. tlv

The "tlv" data type encodes a type-length-value, as defined in [RFC6929] Section 2.3.

Name

tlv

Number

13

Length

Three or more octets

Format


```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  TLV-Type   |  TLV-Length   |  TLV-Data ...  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Subfields

TLV-Type

This field is one octet. Up-to-date values of this field are specified according to the policies and rules described in [RFC6929] Section 10. Values 254-255 are "Reserved" for use by future extensions to RADIUS. The value 26 has no special meaning, and MUST NOT be treated as a Vendor Specific attribute.

The TLV-Type is meaningful only within the context defined by "Type" fields of the encapsulating Attributes, using the dotted-number notation introduced in [RFC6929].

A RADIUS server MAY ignore Attributes with an unknown "TLV-Type".

A RADIUS client MAY ignore Attributes with an unknown "TLV-Type".

A RADIUS proxy SHOULD forward Attributes with an unknown "TLV-Type" verbatim.

TLV-Length

The TLV-Length field is one octet, and indicates the length of this TLV including the TLV-Type, TLV-Length and TLV-Value fields. It MUST have a value between 3 and 255. If a client or server receives a TLV with an invalid TLV-Length, then the attribute which encapsulates that TLV MUST be considered to be an "invalid attribute", and handled as per [RFC6929] Section 2.8.

TLVs having TLV-Length of zero (0) MUST NOT be sent; omit the entire TLV instead.

TLV-Data

The TLV-Data field is one or more octets and contains information specific to the Attribute. The format and length of the TLV-Data field is determined by the TLV-Type and TLV-

Length fields.

The TLV-Data field MUST contain only known RADIUS data types. The TLV-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

2.14. vsa

The "vsa" data type encodes Vendor-Specific data, as given in [RFC2865] Section 5.26. It is used only in the Attr-Data field of a Vendor-Specific Attribute. It MUST NOT appear in the contents of any other data type.

Name

vsa

Number

14

Length

Five or more octets

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Vendor-Id                             |
+-----+-----+-----+-----+-----+-----+-----+-----+
|  VSA-Data ....                                                             |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Subfields

Vendor-Id

The 4 octets are the Network Management Private Enterprise Code [PEN] of the Vendor in network byte order.

VSA-Data

The VSA-Data field is one or more octets. The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished

octets.

The codification of the range of allowed usage of this field is outside the scope of this specification.

It SHOULD be encoded as a sequence of "tlv" fields. The interpretation of the TLV-Type and TLV-Data fields are dependent on the vendor's definition of that attribute.

The "vsa" data type MUST be used as contents of the Attr-Data field of the Vendor-Specific attribute. The "vsa" data type MUST NOT appear in the contents of any other data type.

2.15. extended

The "extended" data type encodes the "Extended Type" format, as given in [RFC6929] Section 2.1. It is used only in the Attr-Data field of an Attribute allocated from the "standard space". It MUST NOT appear in the contents of any other data type.

Name

extended

Number

15

Length

Two or more octets

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Extended-Type | Ext-Data ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Subfields

Extended-Type

The Extended-Type field is one octet. Up-to-date values of this field are specified according to the policies and rules described in [RFC6929] Section 10. Unlike the Type field

defined in [RFC2865] Section 5, no values are allocated for experimental or implementation-specific use. Values 241-255 are reserved and MUST NOT be used.

The Extended-Type is meaningful only within a context defined by the Type field. That is, this field may be thought of as defining a new type space of the form "Type.Extended-Type". See [RFC6929] Section 2.5 for additional discussion.

A RADIUS server MAY ignore Attributes with an unknown "Type.Extended-Type".

A RADIUS client MAY ignore Attributes with an unknown "Type.Extended-Type".

Ext-Data

The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry. The Ext-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

The Ext-Data field is one or more octets.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type" to determine the interpretation of the Ext-Data field.

2.16. long-extended

The "long-extended" data type encodes the "Long Extended Type" format, as given in [RFC6929] Section 2.2. It is used only in the Attr-Data field of an Attribute. It MUST NOT appear in the contents of any other data type.

Name

long-extended

Number

16

Length

Three or more octets

Format

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Extended-Type |M| Reserved   | Ext-Data ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Subfields

Extended-Type

This field is identical to the Extended-Type field defined above in Section 2.13.

M (More)

The More field is one (1) bit in length, and indicates whether or not the current attribute contains "more" than 251 octets of data. The More field **MUST** be clear (0) if the Length field has value less than 255. The More field **MAY** be set (1) if the Length field has value of 255.

If the More field is set (1), it indicates that the Ext-Data field has been fragmented across multiple RADIUS attributes. When the More field is set (1), the attribute **MUST** have a Length field of value 255; there **MUST** be an attribute following this one; and the next attribute **MUST** have both the same Type and Extended Type. That is, multiple fragments of the same value **MUST** be in order and **MUST** be consecutive attributes in the packet, and the last attribute in a packet **MUST NOT** have the More field set (1).

That is, a packet containing a fragmented attribute needs to contain all fragments of the attribute, and those fragments need to be contiguous in the packet. RADIUS does not support inter-packet fragmentation, which means that fragmenting an attribute across multiple packets is impossible.

If a client or server receives an attribute fragment with the "More" field set (1), but for which no subsequent fragment can be found, then the fragmented attribute is considered to be an "invalid attribute", and handled as per [RFC6929] Section 2.8.

Reserved

This field is 7 bits long, and is reserved for future use. Implementations **MUST** set it to zero (0) when encoding an

attribute for sending in a packet. The contents SHOULD be ignored on reception.

Future specifications may define additional meaning for this field. Implementations therefore MUST NOT treat this field as invalid if it is non-zero.

Ext-Data

The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry. The Ext-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

The Ext-Data field is one or more octets.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type" to determine the interpretation of the Ext-Data field.

The length of the data MUST be taken as the sum of the lengths of the fragments (i.e. Ext-Data fields) from which it is constructed. Any interpretation of the resulting data MUST occur after the fragments have been reassembled. If the reassembled data does not match the expected format, each fragment MUST be treated as an "invalid attribute", and the reassembled data MUST be discarded.

We note that the maximum size of a fragmented attribute is limited only by the RADIUS packet length limitation. Implementations MUST be able to handle the case where one fragmented attribute completely fills the packet.

2.17. evs

The "evs" data type encodes an "Extended Vendor-Specific" attribute, as given in [RFC6929] Section 2.4. The "evs" data type is used solely to extend the Vendor Specific space. It MAY appear inside of an "extended" or a "long-extended" data type. It MUST NOT appear in the contents of any other data type.

Name

evs

Number

17

Length

Six or more octets

Format

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Vendor-Id                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Vendor-Type   | EVS-Data ....                                         |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Subfields

Vendor-Id

The 4 octets are the Network Management Private Enterprise Code [PEN] of the Vendor in network byte order.

Vendor-Type

The Vendor-Type field is one octet. Values are assigned at the sole discretion of the Vendor.

EVS-Data

The EVS-Data field is one or more octets. It SHOULD encapsulate a previously defined RADIUS data type. Non-standard data types SHOULD NOT be used. We note that the EVS-Data field may be of data type "tlv".

The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets. We recognise that Vendors have complete control over the contents and format of the Ext-Data field, while at the same time recommending that good practices be followed.

Further codification of the range of allowed usage of this field is outside the scope of this specification.

3. Updated Registries

This section defines a new IANA registry for RADIUS data types, and updates the existing RADIUS Attribute Type registry.

3.1. Create a Data Type Registry

This section defines a new RADIUS registry, called "Data Type". Allocation in this registry requires IETF Review. The "Registration Procedures" for this registry are "Standards Action".

The registry contains three columns of data, as follows.

Value

The number of the data type. The value field is an artifact of the registry, and has no on-the-wire meaning.

Description

The name of the data type. The name field is used only for the registry, and has no on-the-wire meaning.

Reference

The specification where the data type was defined.

The initial contents of the registry are as follows.

Value	Description	Reference
-----	-----	-----
1	integer	[RFC2865], TBD
2	enum	[RFC2865], TBD
3	ipv4addr	[RFC2865], TBD
4	time	[RFC2865], TBD
5	text	[RFC2865], TBD
6	string	[RFC2865], TBD
7	concat	TBD
8	ifid	[RFC3162], TBD
9	ipv6addr	[RFC3162], TBD
10	ipv6prefix	[RFC3162], TBD
11	ipv4prefix	[RFC6572], TBD
12	integer64	[RFC6929], TBD
13	tlv	[RFC6929], TBD
14	evs	[RFC6929], TBD
15	extended	[RFC6929], TBD
16	long-extended	[RFC6929], TBD

3.2. Updates to the Attribute Type Registry

This section updates the RADIUS Attribute Type Registry to have a new column, which is inserted in between the existing "Description" and "Reference" columns. The new column is named "Data Type". The contents of that column are the name of a data type, corresponding to the attribute in that row, or blank if the attribute type is unassigned. The name of the data type is taken from the RADIUS Data Type registry, defined above.

The updated registry follows in CSV format.

```
Value,Description,Data Type,Reference
1,User-Name,string,[RFC2865]
2,User-Password,string,[RFC2865]
3,CHAP-Password,string,[RFC2865]
4,NAS-IP-Address,ipv4addr,[RFC2865]
5,NAS-Port,integer,[RFC2865]
6,Service-Type,enum,[RFC2865]
7,Framed-Protocol,enum,[RFC2865]
8,Framed-IP-Address,ipv4addr,[RFC2865]
9,Framed-IP-Netmask,ipv4addr,[RFC2865]
10,Framed-Routing,enum,[RFC2865]
11,Filter-Id,text,[RFC2865]
12,Framed-MTU,integer,[RFC2865]
13,Framed-Compression,enum,[RFC2865]
14>Login-IP-Host,ipv4addr,[RFC2865]
15>Login-Service,enum,[RFC2865]
16>Login-TCP-Port,integer,[RFC2865]
17,Unassigned,,
18,Reply-Message,text,[RFC2865]
19,Callback-Number,text,[RFC2865]
20,Callback-Id,text,[RFC2865]
21,Unassigned,,
22,Framed-Route,text,[RFC2865]
23,Framed-IPX-Network,ipv4addr,[RFC2865]
24,State,string,[RFC2865]
25,Class,string,[RFC2865]
26,Vendor-Specific,vsa,[RFC2865]
27,Session-Timeout,integer,[RFC2865]
28,Idle-Timeout,integer,[RFC2865]
29,Termination-Action,enum,[RFC2865]
30,Called-Station-Id,text,[RFC2865]
31,Calling-Station-Id,text,[RFC2865]
32,NAS-Identifier,text,[RFC2865]
33,Proxy-State,string,[RFC2865]
34>Login-LAT-Service,text,[RFC2865]
35>Login-LAT-Node,text,[RFC2865]
```

36, Login-LAT-Group, string, [RFC2865]
37, Framed-AppleTalk-Link, integer, [RFC2865]
38, Framed-AppleTalk-Network, integer, [RFC2865]
39, Framed-AppleTalk-Zone, text, [RFC2865]
40, Acct-Status-Type, enum, [RFC2866]
41, Acct-Delay-Time, integer, [RFC2866]
42, Acct-Input-Octets, integer, [RFC2866]
43, Acct-Output-Octets, integer, [RFC2866]
44, Acct-Session-Id, text, [RFC2866]
45, Acct-Authentic, enum, [RFC2866]
46, Acct-Session-Time, integer, [RFC2866]
47, Acct-Input-Packets, integer, [RFC2866]
48, Acct-Output-Packets, integer, [RFC2866]
49, Acct-Terminate-Cause, enum, [RFC2866]
50, Acct-Multi-Session-Id, text, [RFC2866]
51, Acct-Link-Count, integer, [RFC2866]
52, Acct-Input-Gigawords, integer, [RFC2869]
53, Acct-Output-Gigawords, integer, [RFC2869]
54, Unassigned, ,
55, Event-Timestamp, time, [RFC2869]
56, Egress-VLANID, integer, [RFC4675]
57, Ingress-Filters, enum, [RFC4675]
58, Egress-VLAN-Name, text, [RFC4675]
59, User-Priority-Table, string, [RFC4675]
60, CHAP-Challenge, string, [RFC2865]
61, NAS-Port-Type, enum, [RFC2865]
62, Port-Limit, integer, [RFC2865]
63, Login-LAT-Port, text, [RFC2865]
64, Tunnel-Type, enum, [RFC2868]
65, Tunnel-Medium-Type, enum, [RFC2868]
66, Tunnel-Client-Endpoint, text, [RFC2868]
67, Tunnel-Server-Endpoint, text, [RFC2868]
68, Acct-Tunnel-Connection, text, [RFC2867]
69, Tunnel-Password, text, [RFC2868]
70, ARAP-Password, string, [RFC2869]
71, ARAP-Features, string, [RFC2869]
72, ARAP-Zone-Access, enum, [RFC2869]
73, ARAP-Security, integer, [RFC2869]
74, ARAP-Security-Data, text, [RFC2869]
75, Password-Retry, integer, [RFC2869]
76, Prompt, enum, [RFC2869]
77, Connect-Info, text, [RFC2869]
78, Configuration-Token, text, [RFC2869]
79, EAP-Message, concat, [RFC2869]
80, Message-Authenticator, string, [RFC2869]
81, Tunnel-Private-Group-ID, text, [RFC2868]
82, Tunnel-Assignment-ID, text, [RFC2868]
83, Tunnel-Preference, integer, [RFC2868]

84, ARAP-Challenge-Response, string, [RFC2869]
85, Acct-Interim-Interval, integer, [RFC2869]
86, Acct-Tunnel-Packets-Lost, integer, [RFC2867]
87, NAS-Port-Id, text, [RFC2869]
88, Framed-Pool, text, [RFC2869]
89, CUI, string, [RFC4372]
90, Tunnel-Client-Auth-ID, text, [RFC2868]
91, Tunnel-Server-Auth-ID, text, [RFC2868]
92, NAS-Filter-Rule, text, [RFC4849]
93, Unassigned, ,
94, Originating-Line-Info, string, [RFC7155]
95, NAS-IPv6-Address, ipv6addr, [RFC3162]
96, Framed-Interface-Id, ifid, [RFC3162]
97, Framed-IPv6-Prefix, ipv6prefix, [RFC3162]
98, Login-IPv6-Host, ipv6addr, [RFC3162]
99, Framed-IPv6-Route, text, [RFC3162]
100, Framed-IPv6-Pool, text, [RFC3162]
101, Error-Cause Attribute, enum, [RFC3576]
102, EAP-Key-Name, string, [RFC4072] [RFC7268]
103, Digest-Response, text, [RFC5090]
104, Digest-Realm, text, [RFC5090]
105, Digest-Nonce, text, [RFC5090]
106, Digest-Response-Auth, text, [RFC5090]
107, Digest-Nextnonce, text, [RFC5090]
108, Digest-Method, text, [RFC5090]
109, Digest-URI, text, [RFC5090]
110, Digest-Qop, text, [RFC5090]
111, Digest-Algorithm, text, [RFC5090]
112, Digest-Entity-Body-Hash, text, [RFC5090]
113, Digest-CNonce, text, [RFC5090]
114, Digest-Nonce-Count, text, [RFC5090]
115, Digest-Username, text, [RFC5090]
116, Digest-Opaque, text, [RFC5090]
117, Digest-Auth-Param, text, [RFC5090]
118, Digest-AKA-Auts, text, [RFC5090]
119, Digest-Domain, text, [RFC5090]
120, Digest-Stale, text, [RFC5090]
121, Digest-HA1, text, [RFC5090]
122, SIP-AOR, text, [RFC5090]
123, Delegated-IPv6-Prefix, ipv6prefix, [RFC4818]
124, MIP6-Feature-Vector, string, [RFC5447]
125, MIP6-Home-Link-Prefix, ipv6prefix, [RFC5447]
126, Operator-Name, text, [RFC5580]
127, Location-Information, string, [RFC5580]
128, Location-Data, string, [RFC5580]
129, Basic-Location-Policy-Rules, string, [RFC5580]
130, Extended-Location-Policy-Rules, string, [RFC5580]
131, Location-Capable, enum, [RFC5580]

132, Requested-Location-Info, enum, [RFC5580]
133, Framed-Management-Protocol, enum, [RFC5607]
134, Management-Transport-Protection, enum, [RFC5607]
135, Management-Policy-Id, text, [RFC5607]
136, Management-Privilege-Level, integer, [RFC5607]
137, PKM-SS-Cert, concat, [RFC5904]
138, PKM-CA-Cert, concat, [RFC5904]
139, PKM-Config-Settings, string, [RFC5904]
140, PKM-Cryptosuite-List, string, [RFC5904]
141, PKM-SAID, text, [RFC5904]
142, PKM-SA-Descriptor, string, [RFC5904]
143, PKM-Auth-Key, string, [RFC5904]
144, DS-Lite-Tunnel-Name, text, [RFC6519]
145, Mobile-Node-Identifier, string, [RFC6572]
146, Service-Selection, text, [RFC6572]
147, PMIP6-Home-LMA-IPv6-Address, ipv6addr, [RFC6572]
148, PMIP6-Visited-LMA-IPv6-Address, ipv6addr, [RFC6572]
149, PMIP6-Home-LMA-IPv4-Address, ipv4addr, [RFC6572]
150, PMIP6-Visited-LMA-IPv4-Address, ipv4addr, [RFC6572]
151, PMIP6-Home-HN-Prefix, ipv6prefix, [RFC6572]
152, PMIP6-Visited-HN-Prefix, ipv6prefix, [RFC6572]
153, PMIP6-Home-Interface-ID, ifid, [RFC6572]
154, PMIP6-Visited-Interface-ID, ifid, [RFC6572]
155, PMIP6-Home-IPv4-HoA, ipv4prefix, [RFC6572]
156, PMIP6-Visited-IPv4-HoA, ipv4prefix, [RFC6572]
157, PMIP6-Home-DHCP4-Server-Address, ipv4addr, [RFC6572]
158, PMIP6-Visited-DHCP4-Server-Address, ipv4addr, [RFC6572]
159, PMIP6-Home-DHCP6-Server-Address, ipv6addr, [RFC6572]
160, PMIP6-Visited-DHCP6-Server-Address, ipv6addr, [RFC6572]
161, PMIP6-Home-IPv4-Gateway, ipv4addr, [RFC6572]
162, PMIP6-Visited-IPv4-Gateway, ipv4addr, [RFC6572]
163, EAP-Lower-Layer, enum, [RFC6677]
164, GSS-Acceptor-Service-Name, text, [RFC7055]
165, GSS-Acceptor-Host-Name, text, [RFC7055]
166, GSS-Acceptor-Service-Specifics, text, [RFC7055]
167, GSS-Acceptor-Realm-Name, text, [RFC7055]
168, Framed-IPv6-Address, ipv6addr, [RFC6911]
169, DNS-Server-IPv6-Address, ipv6addr, [RFC6911]
170, Route-IPv6-Information, ipv6prefix, [RFC6911]
171, Delegated-IPv6-Prefix-Pool, text, [RFC6911]
172, Stateful-IPv6-Address-Pool, text, [RFC6911]
173, IPv6-6rd-Configuration, tlv, [RFC6930]
174, Allowed-Called-Station-Id, text, [RFC7268]
175, EAP-Peer-Id, string, [RFC7268]
176, EAP-Server-Id, string, [RFC7268]
177, Mobility-Domain-Id, integer, [RFC7268]
178, Preauth-Timeout, integer, [RFC7268]
179, Network-Id-Name, string, [RFC7268]

180, EAPoL-Announcement, concat, [RFC7268]
181, WLAN-HESSID, text, [RFC7268]
182, WLAN-Venue-Info, integer, [RFC7268]
183, WLAN-Venue-Language, string, [RFC7268]
184, WLAN-Venue-Name, text, [RFC7268]
185, WLAN-Reason-Code, integer, [RFC7268]
186, WLAN-Pairwise-Cipher, integer, [RFC7268]
187, WLAN-Group-Cipher, integer, [RFC7268]
188, WLAN-AKM-Suite, integer, [RFC7268]
189, WLAN-Group-Mgmt-Cipher, integer, [RFC7268]
190, WLAN-RF-Band, integer, [RFC7268]
191, Unassigned, ,
192-223, Experimental Use, , [RFC3575]
224-240, Implementation Specific, , [RFC3575]
241, Extended-Attribute-1, extended, [RFC6929]
241. {1-25}, Unassigned, ,
241.26, Extended-Vendor-Specific-1, evs, [RFC6929]
241. {27-240}, Unassigned, ,
241. {241-255}, Reserved, , [RFC6929]
242, Extended-Attribute-2, extended, [RFC6929]
242. {1-25}, Unassigned, ,
242.26, Extended-Vendor-Specific-2, evs, [RFC6929]
242. {27-240}, Unassigned, ,
242. {241-255}, Reserved, , [RFC6929]
243, Extended-Attribute-3, extended, [RFC6929]
243. {1-25}, Unassigned, ,
243.26, Extended-Vendor-Specific-3, evs, [RFC6929]
243. {27-240}, Unassigned, ,
243. {241-255}, Reserved, , [RFC6929]
244, Extended-Attribute-4, extended, [RFC6929]
244. {1-25}, Unassigned, ,
244.26, Extended-Vendor-Specific-4, evs, [RFC6929]
244. {27-240}, Unassigned, ,
244. {241-255}, Reserved, , [RFC6929]
245, Extended-Attribute-5, long-extended, [RFC6929]
245. {1-25}, Unassigned, ,
245.26, Extended-Vendor-Specific-5, evs, [RFC6929]
245. {27-240}, Unassigned, ,
245. {241-255}, Reserved, , [RFC6929]
246, Extended-Attribute-6, long-extended, [RFC6929]
246. {1-25}, Unassigned, ,
246.26, Extended-Vendor-Specific-6, evs, [RFC6929]
246. {27-240}, Unassigned, ,
246. {241-255}, Reserved, , [RFC6929]
247-255, Reserved, , [RFC3575]

4. Suggestions for Specifications

We suggest that these data types be used in new RADIUS specifications. Attributes can usually be completely described through their Attribute Type code, name, and data type. The use of "ASCII art" is then limited only to the definition of new data types, and complex data types.

Use of the new extended attributes [RFC6929] makes ASCII art even more problematic. An attribute can be allocated from the standard space, or from one of the extended spaces. This allocation decision is made after the specification has been accepted for publication. That allocation strongly affects the format of the attribute header, making it nearly impossible to create the correct ASCII art prior to final publication. Allocation from the different spaces also changes the value of the Length field, also making it difficult to define it correctly prior to final publication of the document.

The following fields SHOULD be given when defining new attributes:

Description

A description of the meaning and interpretation of the attribute.

Type

The Attribute Type code, given in the "dotted number" notation from [RFC6929]. Specifications can often leave this as "TBD", and request that IANA fill in the allocated values.

Length

A description of the length of the attribute. For attributes of variable length, a maximum length SHOULD be given.

Data Type

One of the named data types from the RADIUS Data Type registry.

Value

A description of any attribute-specific limitations on the values carried by the specified data type. If there are no attribute-specific limitations, then the description of this field can be omitted.

Where the values are limited to a subset of the possible range, valid range(s) MUST be defined.

For attributes of data type "enum", a list of enumerated values and names MUST be given, as with [RFC2865] Section 5.6.

5. Security Considerations

This specification is concerned solely with updates to IANA registries. As such, there are no security considerations with the document itself.

However, the use of inconsistent names and poorly-defined entities in a protocol is problematic. Inconsistencies in specifications can lead to security and interoperability problems in implementations. Further, having one canonical source for the definition of data types means an implementor has fewer specifications to read. The implementation work is therefore simpler, and is more likely to be correct.

The goal of this specification is to reduce ambiguities in the RADIUS protocol, which we believe will lead to more robust and more secure implementations.

6. IANA Considerations

IANA is instructed to create one new registry as described above in Section 3.1. The "TBD" text in that section should be replaced with the RFC number of this document when it is published.

IANA is instructed to update the RADIUS Attribute Type registry, as described above in Section 3.2.

IANA is instructed to require that all allocation requests in the RADIUS Attribute Type Registry contain a "data type" field. That field is required to contain one of the "data type" names contained in the RADIUS Data Type registry.

7. References

7.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March, 1997.

[RFC2865]

Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3629]

Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.

[RFC6158]

DeKok, A., and Weber, G., "RADIUS Design Guidelines", RFC 6158, March 2011.

[RFC6572]

Xia, F., et al, "RADIUS Support for Proxy Mobile IPv6", RFC 6572, June 2012.

7.2. Informative References

[RFC2868]

Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege, M., and I. Goyret, "RADIUS Attributes for Tunnel Protocol Support", RFC 2868, June 2000.

[RFC2869]

Rigney, C., et al, "RADIUS Extensions", RFC 2869, June 2000.

[RFC3162]

Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.

[RFC6929]

DeKok, A., and Lior, A., "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

[PEN]

<http://www.iana.org/assignments/enterprise-numbers>

Acknowledgments

Stuff

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org

Network Working Group
Internet-Draft
Updates: 6613, 6614 (if approved)
Intended status: Experimental
Expires: January 3, 2015

S. Hartman
Painless Security
July 2, 2014

Larger Packets for RADIUS over TCP
draft-ietf-radext-bigger-packets-01.txt

Abstract

The RADIUS over TLS experiment described in RFC 6614 has opened RADIUS to new use cases where the 4096-octet maximum RADIUS packet proves problematic. This specification extends the RADIUS over TCP experiment to permit larger RADIUS packets. This specification compliments other ongoing work to permit fragmentation of RADIUS authorization information. This document registers a new RADIUS code, an action which requires IESG approval.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements notation	3
2. Changes to Packet Processing	3
2.1. Status-Server Considerations	3
3. Forward and backward Compatibility	4
3.1. Rationale	5
3.2. Discovery	6
4. Too Big Response	6
5. Response Length Attribute	6
6. IANA Considerations	7
7. Security Considerations	7
8. References	7
8.1. Normative References	7
8.2. References	8
Author's Address	8

1. Introduction

The Remote Access Dial-In User Server (RADIUS) over TLS [RFC6614] experiment provides strong confidentiality and integrity for RADIUS [RFC2865]. This enhanced security has opened new opportunities for using RADIUS to convey additional authorization information. As an example, [I-D.ietf-abfab-aaa-saml] describes a mechanism for using RADIUS to carry Security Assertion Markup Language (SAML) messages in RADIUS. Many attributes carried in these SAML messages will require confidentiality or integrity such as that provided by TLS.

These new use cases involve carrying additional information in RADIUS packets. The maximum packet length of 4096 octets is proving insufficient for some SAML messages and for other structures that may be carried in RADIUS.

One approach is to fragment a RADIUS message across multiple packets at the RADIUS layer. RADIUS Fragmentation [I-D.ietf-radext-radius-fragmentation] provides a mechanism to split authorization information across multiple RADIUS messages. That mechanism is necessary in order to split authorization information across existing unmodified proxies.

However, there are some significant disadvantages to RADIUS fragmentation. First, RADIUS is a lock-step protocol, and only one fragment can be in transit at a time as part of a given request.

Also, there is no current mechanism to discover the path Maximum Transmission Unit (MTU) across the entire path that the fragment will travel. As a result, fragmentation is likely both at the RADIUS layer and at the transport layer. When TCP is used, much better transport characteristics can be achieved by fragmentation only at the TCP layer. This specification provides a mechanism to achieve these better transport characteristics when TCP is used. As part of this specification, a new RADIUS code is registered.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Changes to Packet Processing

The maximum length of a RADIUS message is increased from 4096 to 65535. A RADIUS Server implementing this specification MUST be able to receive a packet of maximum length. Servers MAY have a maximum size over which they choose to return an error as discussed in Section 4 rather than processing a received packet; this size MUST be at least 4096 octets.

Clients implementing this specification MUST be able to receive a packet of maximum length; that is clients MUST NOT close a TCP connection simply because a large packet is sent over it. Clients MAY include the Response-Length attribute defined in Section 5 to indicate the maximum size of a packet that they can successfully process. Clients MAY silently discard a packet greater than some configured size; this size MUST be at least 4096 octets. Clients MUST NOT retransmit an unmodified request whose response is larger than the client can process as subsequent responses will likely continue to be too large.

Proxies SHOULD be able to process and forward packets of maximum length. Proxies MUST include the Response-Length attribute when forwarding a request received over a transport with a 4096-octet maximum length over a transport with a higher maximum length.

2.1. Status-Server Considerations

This section extends processing of Status-Server messages as described in section 4.1 and 4.2 of [RFC5997].

Clients implementing this specification SHOULD include the Response-Length attribute in Status-Request messages. Servers are already required to ignore unknown attributes received in this message. by

including the attribute, the client indicates how large of a response it can process to its Status-Server request. It is very unlikely that a response to Status-Server is greater than 4096 octets. However the client also indicates support for this specification which triggers server behavior below.

If a server implementing this specification receives a Response-Length attribute in a Status-Server request, it MUST include a Response-Length attribute indicating the maximum size request it can process in its response to the Status-Server request.

3. Forward and backward Compatibility

An implementation of [RFC6613] will silently discard any packet larger than 4096 octets and will close the TCP connection. This section provides guidelines for interoperability with these implementations. These guidelines are stated at the SHOULD level. In some environments support for large packets will be important enough that roaming or other agreements will mandate their support. In these environments, all implementations might be required to support this specification removing the need for interoperability with RFC 6613. It is likely that these guidelines will be relaxed to the MAY level and support for this specification made a requirement if RADIUS over TLS and TCP are moved to the standards track in the future.

Clients SHOULD provide configuration for the maximum size of a request sent to each server. Servers SHOULD provide configuration for the maximum size of a response sent to each client. If dynamic discovery mechanisms are supported, configuration SHOULD be provided for the maximum size of clients and servers in each dynamic discovery category.

If a client sends a request larger than 4096 octets and the TCP connection is closed without a response, the client SHOULD treat the request as if a request too big error (Section 4) specifying a maximum size of 4096 is received. Clients or proxies sending multiple requests over a single TCP connection without waiting for responses SHOULD implement capability discovery as discussed in Section 3.2.

By default, a server SHOULD not generate a response larger than 4096 octets. The Response-Length attribute MAY be included in a request to indicate that larger responses are desirable. Other attributes or configuration MAY be used as an indicator that large responses are likely to be acceptable.

A proxy that implements both this specification and RADIUS Fragmentation [I-D.ietf-radext-radius-fragmentation] SHOULD use RADIUS fragmentation when the following conditions are met:

1. A packet is being forwarded towards an endpoint whose configuration does not support a packet that large.
2. RADIUS Fragmentation can be used for the packet in question.

3.1. Rationale

The interoperability challenge appears at first significant. This specification proposes to introduce behavior where new implementations will fail to function with existing implementations.

However, these capabilities are introduced to support new use cases. If an implementation has 10000 octets of attributes to send, it cannot in general trim down the response to something that can be sent. Under this specification a large packet would be generated that will be silently discarded by an existing implementation. Without this specification, no packet is generated because the required attributes cannot be sent.

The biggest risk to interoperability would be if requests and responses are expanded to include additional information that is not strictly necessary. So, avoiding creating situations where large packets are sent to existing implementations is mostly an operational matter. Interoperability is most impacted when the size of packets in existing use cases is significantly increased and least impacted when large packets are used for new use cases where the deployment is likely to require updated RADIUS implementations.

There is a special challenge for proxies or clients with high request volume. When an RFC 6113 implementation receives a packet that is too large, it closes the connection and does not respond to any requests in process. Such a client would lose requests and might find distinguishing request-too-big situations from other failures difficult. In these cases, the discovery mechanism described in Section 3.2 can be used.

Also, RFC 6613 is an experiment. Part of running that experiment is to evaluate whether additional changes are required to RADIUS. A lower bar for interoperability should apply to changes to experimental protocols than standard protocols.

This specification provides good facilities to enable implementations to understand packet size when proxying to/from standards-track UDP RADIUS.

3.2. Discovery

As discussed in Section 2.1, a client MAY send a Status-Server message to discover whether an authentication or accounting server supports this specification. The client includes a Response-Length attribute; this signals the server to include a Response-Length attribute indicating the maximum packet size the server can process. In this one instance, Response-Length indicate the size of a request that can be processed rather than a response.

4. Too Big Response

When a RADIUS server receives a request that is larger than can be processed, it generates a Too-Big response as follows:

The code is TBDCODE.

The identifier is the identifier from the request.

The Response-Length attribute MUST be included and its value is the maximum size of request that will be processed.

No other attributes SHOULD be included. Additional attributes MUST be ignored by the client.

Clients will not typically be able to adjust and resend requests when this error is received. In some cases the client can fall back to RADIUS Fragmentation. In other cases this code will provide for better client error reporting and will avoid retransmitting requests guaranteed to fail.

5. Response Length Attribute

The following RADIUS attribute type values [RFC3575] are assigned. The assignment rules in section 10.3 of [RFC6929] are used.

Name	Attribute	Description
Response-Length	TBD	2-octet unsigned integer maximum response length

The Response-Length attribute MAY be included in any RADIUS request. In this context it indicates the maximum length of a response the client is prepared to receive. Values are between 4096 and 65535. The attribute MAY also be included in a response to a Status-Server message. In this case the attribute indicate the maximum size RADIUS request that is permitted.

6. IANA Considerations

A new RADIUS packet type code is registered in the RADIUS packet type codes registry discussed in section 2.1 of RFC 3575 [RFC3575]. The name is "Too-Big" and the code is TBDCODE. The IESG is requested to approve this registration along with approving publication of this document.

IANA is requested to assign the RADIUS type defined in section Section 5

7. Security Considerations

This specification updates RFC 6613 and will be used with [RFC6614]. When used over plain TCP, this specification creates new opportunities for an on-path attacker to impact availability. these attacks can be entirely mitigated by using TLS.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003.
- [RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, August 2010.
- [RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, May 2012.

- [RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.
- [RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

8.2. References

- [I-D.ietf-abfab-aaa-saml]
Howlett, J. and S. Hartman, "A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for SAML", draft-ietf-abfab-aaa-saml-09 (work in progress), February 2014.
- [I-D.ietf-radext-radius-fragmentation]
Perez-Mendez, A., Lopez, R., Pereniguez-Garcia, F., Lopez-Millan, G., Lopez, D., and A. DeKok, "Support of fragmentation of RADIUS packets", draft-ietf-radext-radius-fragmentation-06 (work in progress), April 2014.

Author's Address

Sam Hartman
Painless Security
Email: hartmans-ietf@mit.edu

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 14, 2014

D. Cheng
Huawei
J. Korhonen
Broadcom
M. Boucadair
France Telecom
S. Sivakumar
Cisco Systems
June 12, 2014

RADIUS Extensions for IP Port Configuration and Reporting
draft-ietf-radext-ip-port-radius-ext-01

Abstract

This document defines three new RADIUS attributes. For devices that implementing IP port ranges, these attributes are used to communicate with a RADIUS server in order to configure and report TCP/UDP ports and ICMP identifiers, as well as mapping behavior for specific hosts. This mechanism can be used in various deployment scenarios such as CGN (Carrier Grade NAT), NAT64, Provider WLAN Gateway, etc.

This document does not make any assumption about the deployment context.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 14, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Extensions of RADIUS Attributes and TLVs	5
3.1. Extended Attributes for IP Ports	6
3.1.1. Extended-Type and IP-Port-Type TLV	6
3.1.2. IP-Port-Limit Attribute	7
3.1.3. IP-Port-Range Attribute	9
3.1.4. IP-Port-Forwarding-Map Attribute	12
3.2. RADIUS TLVs for IP Ports	14
3.2.1. IP-Port-Limit TLV	14
3.2.2. IP-Port-Ext-IPv4-Addr TLV	15
3.2.3. IP-Port-Int-IP-Addr TLV	16
3.2.4. IP-Port-Int-Port TLV	17
3.2.5. IP-Port-Ext-Port TLV	17
3.2.6. IP-Port-Alloc TLV	18
3.2.7. IP-Port-Range-Start TLV	19
3.2.8. IP-Port-Range-End TLV	20
3.2.9. IP-Port-Local-Id TLV	20
4. Applications, Use Cases and Examples	21
4.1. Managing CGN Port Behavior using RADIUS	21
4.1.1. Configure IP Port Limit for a User	22
4.1.2. Report IP Port Allocation/De-allocation	24
4.1.3. Configure Forwarding Port Mapping	25
4.1.4. An Example	27
4.2. Report Assigned Port Set for a Visiting UE	28
5. Table of Attributes	29
6. Security Considerations	30
7. IANA Considerations	30
8. Acknowledgements	31
9. References	31
9.1. Normative References	31

9.2. Informative References	32
Authors' Addresses	33

1. Introduction

In a broadband network, customer information is usually stored on a RADIUS server [RFC2865] and at the time when a user initiates an IP connection request, the RADIUS server will populate the user's configuration information to the Network Access Server (NAS), which is usually co-located with the Border Network Gateway (BNG), after the connection request is granted. The Carrier Grade NAT (CGN) function may also be implemented on the BNG, and therefore CGN TCP/UDP port (or ICMP identifier) mapping behavior can be configured on the RADIUS server as part of the user profile, and populated to the NAS in the same manner. In addition, during the operation, the CGN can also convey port/identifier mapping behavior specific to a user to the RADIUS server, as part of the normal RADIUS accounting process.

The CGN device that communicates with a RADIUS server using RADIUS extensions defined in this document may perform NAT44 [RFC3022], NAT64 [RFC6146], or Dual-Stack Lite AFTR [RFC6333] function.

For the CGN case, when IP packets traverse a CGN device, it would perform TCP/UDP source port mapping or ICMP identifier mapping as required. A TCP/UDP source port or ICMP identifier, along with source IP address, destination IP address, destination port and protocol identifier if applicable, uniquely identify a session. Since the number space of TCP/UDP ports and ICMP identifiers in CGN's external realm is shared among multiple users assigned with the same IPv4 address, the total number of a user's simultaneous IP sessions is likely to be subject to port quota (see Section 5 of [RFC6269]).

The attributes defined in this document may also be used to report the assigned port range in some deployments such as Provider WLAN [I-D.gundavelli-v6ops-community-wifi-svcs]. For example, a visiting host can be managed by a CPE (Customer Premises Equipment) which will need to report the assigned port range to the service platform. This is required for identification purposes (see WT-146 for example).

This document proposes three new attributes as RADIUS protocol's extensions, and they are used for separate purposes as follows:

1. IP-Port-Limit: This attribute may be carried in RADIUS Access-Accept, Access-Request, Accounting-Request or CoA-Request packet. The purpose of this attribute is to limit the total number of

TCP/UDP ports and/or ICMP identifiers that an IP subscriber can use, associated with an IPv4 address.

2. IP-Port-Range: This attribute may be carried in RADIUS Accounting-Request packet. The purpose of this attribute is to report by an address sharing device (e.g., a CGN) to the RADIUS server the range of TCP/UDP ports and/or ICMP identifiers that have been allocated or deallocated associated with a given IPv4 address for a subscriber.
3. IP-Port-Forwarding-Map: This attribute may be carried in RADIUS Access-Accept, Access-Request, Accounting-Request or CoA-Request packet. The purpose of this attribute is to specify how a TCP/UDP port (or an ICMP identifier) mapping to another TCP/UDP port (or an ICMP identifier), and each is associated with its respective IPv4 address.

This document was constructed using the [RFC2629] .

2. Terminology

This document makes use if the following terms:

- o IP Port: refers to the port numbers of IP transport protocols, including TCP port, UDP port and ICMP identifier.
- o IP Port Type: refers to one of the following: (1)TCP/UDP port and ICMP identifier, (2)TCP port and UDP port, (3) TCP port, (4) UDP port, or (5)ICMP identifier.
- o IP Port Limit: denotes the maximum number of IP ports for a specific port type, that a device supporting port ranges can use when performing port number mapping for a specific user.
- o IP Port Range: specifies a set of contiguous IP ports, indicated by the smallest numerical number and the largest numerical number, inclusively.
- o Internal IP Address: refers to the IP address that is used as a source IP address in an outbound IP packet sent towards a device supporting port ranges in the internal realm. In the IPv4 case, it is typically a private address [RFC1918].
- o External IP Address: refers to the IP address that is used as a source IP address in an outbound IP packet after traversing a device supporting port ranges in the external realm. In the IPv4 case, it is typically a global routable IP address.

- o Internal Port: is a UDP or TCP port, or an ICMP identifier, which is allocated by a host or application behind a device supporting port ranges for an outbound IP packet in the internal realm.
- o External Port: is a UDP or TCP port, or an ICMP identifier, which is allocated by a device supporting port ranges upon receiving an outbound IP packet in the internal realm, and is used to replace the internal port that is allocated by a user or application.
- o External realm: refers to the networking segment where IPv4 public addresses are used in respective of the device supporting port ranges.
- o Internal realm: refers to the networking segment that is behind a device supporting port ranges and where IPv4 private addresses are used.
- o Mapping: associates with a device supporting port ranges for a relationship between an internal IP address, internal port and the protocol, and an external IP address, external port, and the protocol.
- o Port-based device: a device that is capable of providing IP address and IP port mapping services and in particular, with the granularity of one or more subsets within the 16-bit IP port number range. A typical example of this device is a CGN, CPE, Provider WLAN Gateway, etc.

Note the terms "internal IP address", "internal port", "internal realm", "external IP address", "external port", "external realm", and "mapping" and their semantics are the same as in [RFC6887], and [RFC6888].

3. Extensions of RADIUS Attributes and TLVs

These three new attributes are defined in the following sub-sections:

1. IP-Port-Limit Attribute
2. IP-Port-Range Attribute
3. IP-Port-Forwarding-Map Attribute

All these attributes are allocated from the RADIUS "Extended Type" code space per [RFC6929].

3.1. Extended Attributes for IP Ports

3.1.1. Extended-Type and IP-Port-Type TLV

This section defines a new Extended-Type and an IP-Port-Type TLV (see Figure 1).

The IP port type may be one of the following:

- o TCP port, UDP port, and ICMP identifier
- o TCP port and UDP port
- o TCP port
- o UDP port
- o ICMP identifier

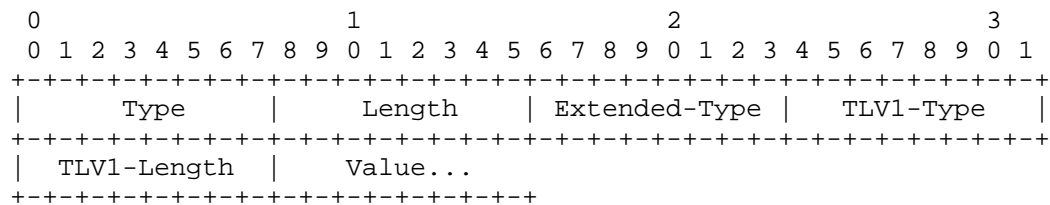


Figure 1

Type:

TBA1 - Extended-Type-1 (241), Extended-Type-2 (242), Extended-Type-3 (243), or Extended-Type-4 (244) per [RFC6929].

Length:

This field indicates the total length in bytes of all fields this attribute, including the Type, Length, Extended-Type, and the embedded TLVs.

Extended-Type:

TBA2.

TLV1-Type:

Type field of IP-Port-Type TLV. This one byte field indicates the IP port type as follows:

TBA2-1:

Refer to TCP port, UDP port, and ICMP identifier as a whole.

TBA2-2:

Refer to TCP port and UDP port as a whole.

TBA2-3:

Refer to TCP port only.

TBA2-4:

Refer to UDP port only.

TBA2-5:

Refer to ICMP identifier only.

TLV1-Length:

Length field of IP-Port-Type TLV. This field indicates the total length in bytes of the TLV1, including the field of TLV1-Type, TLV1-Length, and the Value.

Value:

Value field of IP-Port-Type TLV. This field contains one or more TLVs, refer to Section 3.1.2, Section 3.1.3, Section 3.1.4 for details.

The interpretation of this field is determined by the identifier of "TBA1.TBA2.{TBA2-1..TBA2-5}" along with the embedded TLVs.

3.1.2. IP-Port-Limit Attribute

This attribute contains the Extended-Type and IP-Port-Type TLV defined in Section 3.1.1, along with the embedded IP-Port-Limit TLV and IP-Port-Ext-IPv4-Addr TLV, defined in Section 3.2.1 and Section 3.2.2, respectively. It specifies the maximum number of IP ports, as indicated in IP-Port-Limit TLV, of a specific port type, and associated with a given IPv4 address, as indicated in IP-Port-Ext-IPv4-Addr TLV for an end user. Note that when IP-Port-Ext-IPv4-Addr TLV is not included as part of the IP-Port-Limit Attribute,

the port limit is applied to all the IPv4 addresses managed by the port device, e.g., a CGN or NAT64 device.

The IP-Port-Limit Attribute MAY appear in an Access-Accept packet. It MAY also appear in an Access-Request packet as a hint by the device supporting port ranges, which is co-allocated with the NAS, to the RADIUS server as a preference, although the server is not required to honor such a hint.

The IP-Port-Limit Attribute MAY appear in a CoA-Request packet.

The IP-Port-Limit Attribute MAY appear in an Accounting-Request packet.

The IP-Port-Limit Attribute MUST NOT appear in any other RADIUS packets.

The format of the IP-Port-Limit Attribute is shown in Figure 2. The fields are transmitted from left to right.

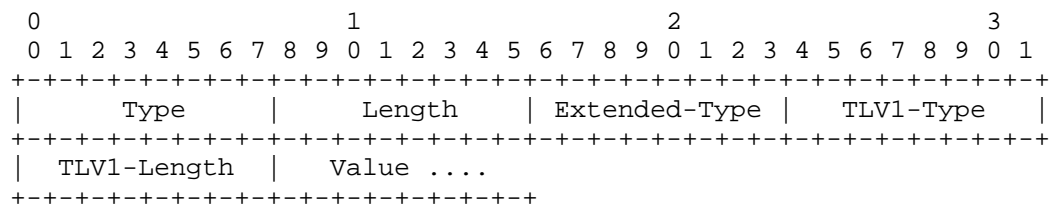


Figure 2

Type:

TBA1 - Extended-Type-1 (241), Extended-Type-2 (242), Extended-Type-3 (243), or Extended-Type-4 (244) per [RFC6929].

Length:

This field indicates the total length in bytes of all fields of this attribute, including the Type, Length, Extended-Type, and the entire length of the embedded TLVs.

Extended-Type:

TBA2 - This one byte field contains a value that indicates the IP port type, refer to Section 3.1.1 for detail.

TLV1-Type:

TBA2-1, TBA2-2, TBA2-3, TBA2-4, or TBA2-5. Refer to Section 3.1.1 for detail.

TLV1-Length:

This field indicates the total length in bytes of the TLV1, including the field of TLV1-Type, TLV1-Length, and the entire length of the embedded TLVs.

Value:

This field contains a set of TLVs as follows:

IP-Port-Limit TLV:

This TLV contains the maximum number of IP ports of a specific IP port type and associated with a given IPv4 address for an end user. This TLV must be included in the IP-Port-Limit Attribute. Refer to Section 3.2.1.

IP-Port-Ext-IPv4-Addr TLV:

This TLV contains the IPv4 address that is associated with the IP port limit contained in the IP-Port-Limit TLV. This TLV is optionally included as part of the IP-Port-Limit Attribute. Refer to Section 3.2.2.

IP-Port-Limit attribute is associated with the following identifier: Type(TBA1).Extended-Type(TBA2).IP-Port-Type TLV{TBA2-1..TBA2-5}.[IP-Port-Limit TLV(TBA3), {IP-Port-Ext-IPv4-Addr TLV (TBA4)}].

3.1.3. IP-Port-Range Attribute

This attribute contains the Extended-Type and IP-Port-Type TLV defined in Section 3.1.1, along with a set of embedded TLVs defined in Section 3.2.7 (IP-Port-Range-Start TLV), Section 3.2.8 (IP-Port-Range-End TLV), Section 3.2.6 (IP-Port-Alloc TLV), Section 3.2.2 (IP-Port-Ext-IPv4-Addr TLV), and Section 3.2.9 (IP-Port-Local-Id TLV). It contains a range of contiguous IP ports of a specific port type and associated with an IPv4 address that are either allocated or deallocated by a device for a given subscriber, and the information is intended to send to RADIUS server.

This attribute can be used to convey a single IP port number; in such case IP-Port-Range-Start and IP-Port-Range-End conveys the same value.

Within an IP-Port-Range Attribute, the IP-Port-Alloc TLV is always included. For port allocation, both IP-Port-Range-Start TLV and IP-Port-Range-End TLV must be included; for port deallocation, the inclusion of these two TLVs is optional and if not included, it implies that all ports that are previously allocated are now deallocated. Both IP-Port-Ext-IPv4-Addr TLV and IP-Port-Local-Id TLV are optional and if included, they are used by a port device (e.g., a CGN device) to identify the end user.

The IP-Port-Range Attribute MAY appear in an Accounting-Request packet.

The IP-Port-Range Attribute MUST NOT appear in any other RADIUS packets.

The format of the IP-Port-Range Attribute format is shown in Figure 3. The fields are transmitted from left to right.

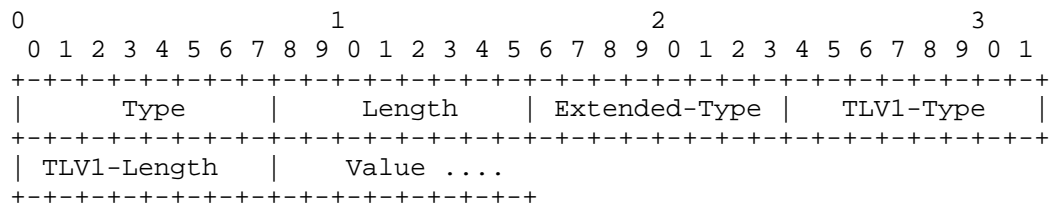


Figure 3

Type:

TBA1 - Extended-Type-1 (241), Extended-Type-2 (242), Extended-Type-3 (243), or Extended-Type-4 (244) per [RFC6929]

Length:

This field indicates the total length in bytes of all fields of this attribute, including the Type, Length, Extended-Type, and the entire length of the embedded TLVs.

Extended-Type:

TBA2 - This one byte field contains a value that indicates the IP port type, refer to Section 3.1.1 for detail.

TLV1-Type:

TBA2-1, TBA2-2, TBA2-3, TBA2-4, or TBA2-5. Refer to Section 3.1.1 for detail.

TLV1-Length:

This field indicates the total length in bytes of the TLV1, including the field of TLV1-Type, TLV1-Length, and the entire length of the embedded TLVs.

Value:

This field contains a set of TLVs as follows:

IP-Port-Alloc TLV:

This TLV contains a flag to indicate that the range of the specified IP ports for either allocation or deallocation. This TLV must be included as part of the IP-Port-Range Attribute. Refer to Section 3.2.6.

IP-Port-Range-Start TLV:

This TLV contains the smallest port number of a range of contiguous IP ports. To report the port allocation, this TLV must be included together with IP-Port-Range-End TLV as part of the IP-Port-Range Attribute. Refer to Section 3.2.7.

IP-Port-Range-End TLV:

This TLV contains the largest port number of a range of contiguous IP ports. To report the port allocation, this TLV must be included together with IP-Port-Range-Start TLV as part of the IP-Port-Range Attribute. Refer to Section 3.2.8.

IP-Port-Ext-IPv4-Addr TLV:

This TLV contains the IPv4 address that is associated with the IP port range, as collectively indicated in the IP-Port-Range-Start TLV and the IP-Port-Range-End TLV. This TLV is optionally included as part of the IP-Port-Range Attribute. Refer to Section 3.2.2.

IP-Port-Local-Id TLV:

This TLV contains a local session identifier at the customer premise, such as MAC address, interface ID, VLAN ID, PPP sessions ID, VRF ID, IPv6 address/prefix, etc. This TLV is

optionally included as part of the IP-Port-Range Attribute.
Refer to Section 3.2.9.

The IP-Port-Range attribute is associated with the following identifier: Type(TBA1).Extended-Type(TBA2).IP-Port-Type TLV{ TBA2-1..TBA2-5}.[IP-Port-Alloc TLV(TBA8), {IP-Port-Range-Start TLV (TBA9), IP-Port-Range-End TLV (TBA10)}, {IP-Port-Ext-IPv4-Addr TLV (TBA4)}, {IP-Port-Local-Id TLV (TBA11)}].

3.1.4. IP-Port-Forwarding-Map Attribute

This attribute contains the Extended-Type and IP-Port-Type TLV defined in Section 3.1.1, along with a set of embedded TLVs defined in Section 3.2.4 (IP-Port-Int-Port TLV), Section 3.2.5 (IP-Port-Ext-Port TLV), Section 3.2.3 (IP-Port-Int-IP-Addr TLV), Section 3.2.9 (IP-Port-Local-Id TLV) and Section 3.2.2 (IP-Port-Ext-IP-Addr TLV). The attribute contains a 2-byte IP internal port number that is associated with an internal IPv4 or IPv6 address, or a locally significant identifier at the customer site, and a 2-byte IP external port number that is associated with an external IPv4 address. The internal IPv4 or IPv6 address, or the local identifier must be included; the external IPv4 address may also be included.

The IP-Port-Forwarding-Map Attribute MAY appear in an Access-Accept packet. It MAY also appear in an Access-Request packet as a hint by the device supporting port mapping, which is co-allocated with the NAS, to the RADIUS server as a preference, although the server is not required to honor such a hint.

The IP-Port-Forwarding-Map Attribute MAY appear in a CoA-Request packet.

The IP-Port-Forwarding-Map Attribute MAY also appear in an Accounting-Request packet.

The attribute MUST NOT appear in any other RADIUS packet.

The format of the IP-Port-Forwarding-Map Attribute is shown in Figure 4. The fields are transmitted from left to right.

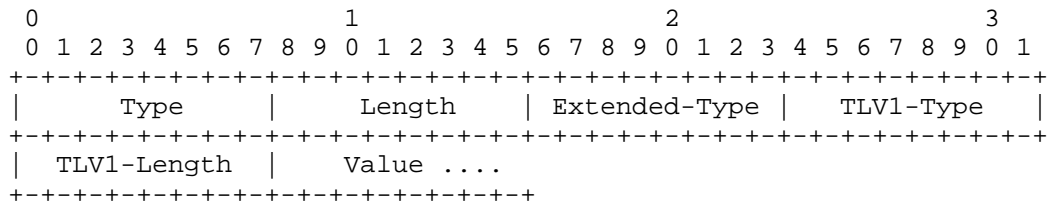


Figure 4

Type:

TBA1 - Extended-Type-1 (241), Extended-Type-2 (242), Extended-Type-3 (243), or Extended-Type-4 (244) per [RFC6929]

Length:

This field indicates the total length in bytes of all fields of this attribute, including the Type, Length, Extended-Type, and the entire length of the embedded TLVs.

Extended-Type:

This one byte field contains a value that indicates the IP port type, refer to Section 3.1.1 for details.

TLV1-Type:

TBA2-1, TBA2-2, TBA2-3, TBA2-4, or TBA2-5. Refer to Section 3.1.1 for detail.

TLV1-Length:

This field indicates the total length in bytes of the TLV1, including the field of TLV1-Type, TLV1-Length, and the entire length of the embedded TLVs.

Value:

This field contains a set of TLVs as follows:

IP-Port-Int-Port TLV:

This TLV contains an internal IP port number associated with an internal IPv4 or IPv6 address. This TLV must be included together with IP-Port-Ext-Port TLV as part of the IP-Port-Forwarding-Map attribute. Refer to Section 3.2.4.

IP-Port-Ext-Port TLV:

This TLV contains an external IP port number associated with an external IPv4 address. This TLV must be included together with IP-Port-Int-Port TLV as part of the IP-Port-Forwarding-Map attribute. Refer to Section 3.2.5.

IP-Port-Int-IP-Addr TLV:

This TLV contains an IPv4 or IPv6 address that is associated with the internal IP port number contained in the IP-Port-Int-Port TLV. Either this TLV or IP-Port-Local-Id TLV must be included as part of the IP-Port-Forwarding-Map Attribute. Refer to Section 3.2.3.

IP-Port-Local-Id TLV:

This TLV contains a local session identifier at the customer premise, such as MAC address, interface ID, VLAN ID, PPP sessions ID, VRF ID, IPv6 address/prefix, etc. Either this TLV or IP-Port-Int-IP-Addr TLV must be included as part of the IP-Port-Forwarding-Map Attribute. Refer to Section 3.2.9.

IP-Port-Ext-IPv4-Addr TLV:

This TLV contains an IPv4 address that is associated with the external IP port number contained in the IP-Port-Ext-Port TLV. This TLV may be included as part of the IP-Port-Forwarding-Map Attribute. Refer to Section 3.2.2.

The IP-Port-Forwarding-Map attribute is associated with the following identifier: Type(TBA1).Extended-Type(TBA2).IP-Port-Type TLV{TBA2-1..TBA2-5}.[IP-Port-Int-Port TLV(TBA6), IP-Port-Ext-Port TLV(TBA7), {IP-Port-Int-IP-Addr TLV (TBA5)}, {IP-Port-Ext-IPv4-Addr TLV (TBA4)}].

3.2. RADIUS TLVs for IP Ports

3.2.1. IP-Port-Limit TLV

This TLV (Figure 5) uses the format defined in [RFC6929]. Its Value field contains a 2-byte integer called IP-Port-Limit, which indicates the maximum number of ports of a specified IP-Port-Type and associated with a given IPv4 address assigned to a subscriber.

IP-Port-Limit TLV is included as part of the IP-Port-Limit Attribute (refer to Section 3.1.2).

Note that IP-Port-Limit TLV is embedded within IP-Port-Type TLV (refer to Section 3.1.1) for detail.

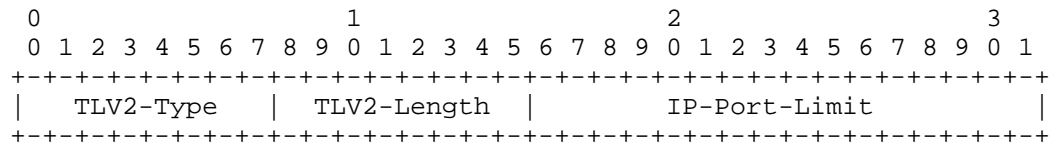


Figure 5

TLV2-Type:

TBA3: The type field for IP-Port-Limit TLV.

TLV2-Length:

This field indicates the total length in bytes of the TLV2, including the field of TLV2-Type, TLV2-Length, and the Value field, i.e., IP-Port-Limit.

IP-Port-Limit:

2-byte integer. This field contains the maximum number of IP ports of which, the port type is specified by container IP-Port-Type TLV.

3.2.2. IP-Port-Ext-IPv4-Addr TLV

This TLV (Figure 6) uses the format defined in[RFC6929]. Its Value field contains a 4-byte External IPv4 address.

IP-Port-Ext-IPv4-Addr TLV can be included as part of the IP-Port-Limit Attribute (refer to Section 3.1.2), IP-Port-Range Attribute (refer to Section 3.1.3), and IP-Port-Forwarding-Map Attribute (refer to Section 3.1.4).

Note that IP-Port-Ext-IPv4-Addr TLV is embedded within IP-Port-Type TLV (refer to Section 3.1.1) for detail.

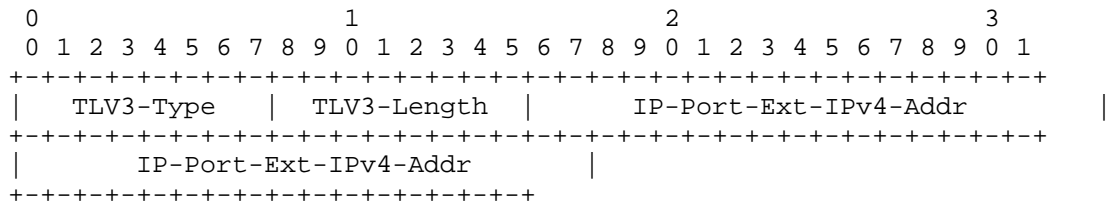


Figure 6

TLV3-Type:

TBA4: The type field for IP-Port-IPv4-Addr TLV.

TLV3-Length:

6. The Length field for IP-Port-IPv4-Addr TLV.

IP-Port-Ext-IPv4-Addr:

4-byte integer. This field contains the IPv4 address that is associated with the range of IP ports.

3.2.3. IP-Port-Int-IP-Addr TLV

This TLV (Figure 7) uses format defined in [RFC6929]. Its Value field contains an internal IPv4 or IPv6 address.

IP-Port-Int-IP-Addr TLV can be included as part of the IP-Port-Forwarding-Map Attribute (refer to Section 3.1.4).

Note that IP-Port-Int-IP-Addr TLV is embedded within IP-Port-Type TLV (refer to Section 3.1.1) for detail.

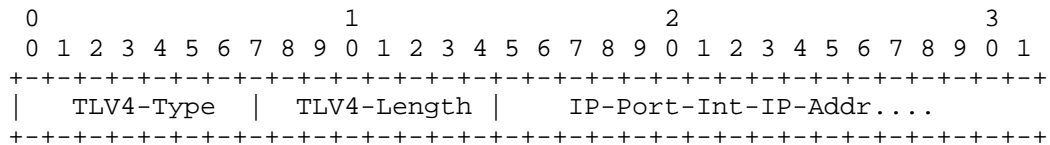


Figure 7

TLV4-Type:

TBA5: The type field for IP-Port-Int-IP-Addr TLV.

TLV4-Length:

6 or 18 bytes. The Length field for IP-Port-Int-IP-Addr TLV.

IP-Port-Int-IP-Addr:

4 byte integer for IPv4 address or 16 byte for IPv6 address.

3.2.4. IP-Port-Int-Port TLV

This TLV (Figure 8) uses format defined in [RFC6929]. Its Value field contains an internal IP port number that is associated with an internal IPv4 or IPv6 address.

IP-Port-Int-Port TLV is included as part of the IP-Port-Forwarding-Map Attribute (refer to Section 3.1.4).

IP-Port-Int-Port TLV is embedded within embedded within IP-Port-Type TLV (refer to Section 3.1.1) for detail.

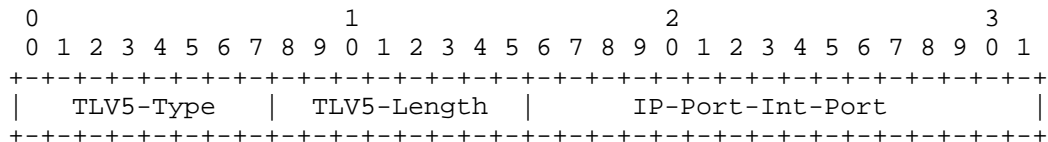


Figure 8

TLV5-Type:

TBA6: The type field for IP-Port-Int-Port TLV.

TLV5-Length:

4 bytes. The Length field for IP-Port-Int-Port TLV.

IP-Port-Int-Port:

2 byte integer. The internal IP port number that is associated with an IPv4 or IPv6 address.

3.2.5. IP-Port-Ext-Port TLV

This TLV (Figure 9) uses format defined in [RFC6929]. Its Value field contains an external IP port number that is associated with an external IPv4 address.

IP-Port-Ext-Port TLV is included as part of the IP-Port-Forwarding-Map Attribute (refer to Section 3.1.4).

IP-Port-Ext-Port TLV is embedded within IP-Port-Type TLV (refer to Section 3.1.1) for detail.

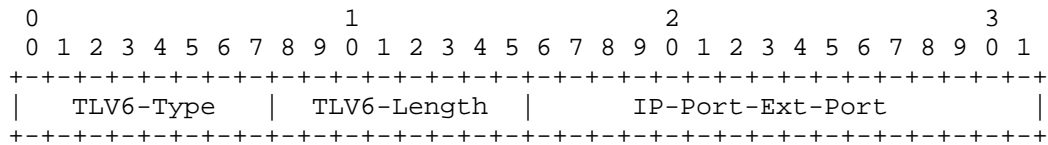


Figure 9

TLV6-Type:

TBA7: The type field for IP-Port-Ext-Port TLV.

TLV6-Length:

4 bytes. The Length field for IP-Port-Ext-Port TLV.

IP-Port-Ext-Port:

2 byte integer. The external IP port number that is associated with an IPv4 address.

3.2.6. IP-Port-Alloc TLV

This TLV (Figure 10) uses format defined in [RFC6929]. Its Value field contains a 2-byte integer called IP-Port-Alloc, which indicates either the allocation or deallocation of a range of IP ports.

IP-Port-Alloc TLV is included as part of the IP-Port-Range Attribute (refer to Section 3.1.3).

Note that IP-Port-Alloc TLV is embedded within IP-Port-Type TLV (refer to Section 3.1.1) for detail.

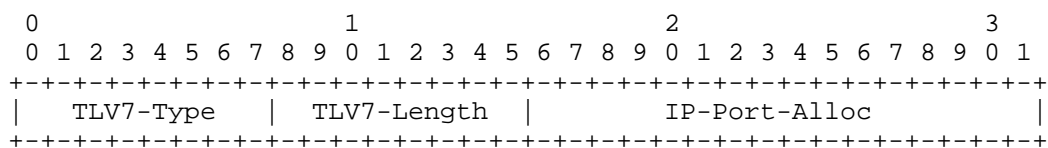


Figure 10

TLV7-Type:

TBA8: The type field for IP-Port-Alloc TLV.

2-byte integer. This field contains the smallest port number of a range of contiguous IP ports.

3.2.8. IP-Port-Range-End TLV

This TLV (Figure 12) uses format defined in [RFC6929]. Its Value field contains a 2-byte integer called IP-Port-Range-End, which indicates largest port number of a range of contiguous IP ports.

IP-Port-Range-End TLV is included as part of the IP-Port-Range Attribute (refer to Section 3.1.3).

Note that IP-Port-Range-End TLV is embedded within IP-Port-Type TLV (refer to Section 3.1.1) for detail.

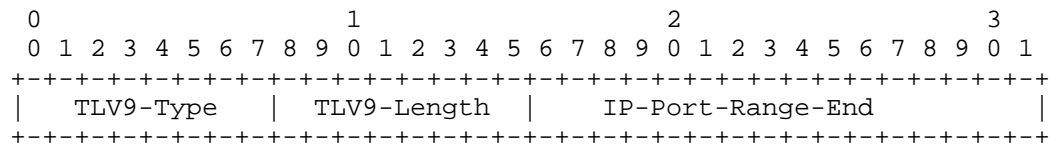


Figure 12

TLV9-Type:

TBA10: The type field for IP-Port-Range-End TLV.

TLV9-Length:

4. The Length field for IP-Port-Range-End TLV.

IP-Port-Range-End:

2-byte integer. This field contains the largest port number of a range of contiguous IP ports.

3.2.9. IP-Port-Local-Id TLV

This TLV (Figure 13) uses format defined in [RFC6929]. Its Value field contains an identifier with local significance.

In some CGN deployment scenarios as described such as L2NAT [I-D.miles-behave-l2nat], DS-Extra-Lite [RFC6619] and Lightweight 4over6 [I-D.ietf-softwire-lw4over6], parameters at a customer premise such as MAC address, interface ID, VLAN ID, PPP session ID, IPv6 prefix, VRF ID, etc., may also be required to pass to the RADIUS server as part of the accounting record.

IP-Port-Local-Id TLV can be included as part of the IP-Port-Range Attribute (refer to Section 3.1.3) and IP-Port-Forwarding-Map Attribute (refer to Section 3.1.4).

Note that IP-Port-Local-Id TLV is embedded within IP-Port-Type TLV (refer to Section 3.1.1) for detail.

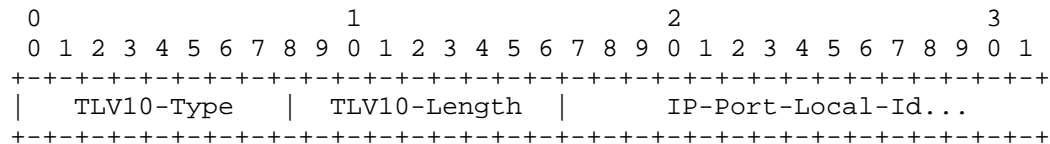


Figure 13

TLV10-Type:

TBA11: The type field for IP-Port-Local-Id TLV.

TLV10-Length:

Variable number of bytes. The Length field for IP-Port-Local-Id TLV.

IP-Port-Local-Id:

This is a local session identifier at the customer premise, such as MAC address, interface ID, VLAN ID, PPP sessions ID, VRF ID, IPv6 address/prefix, etc. The length of this field is the value contained in TLV7-Length field minus 2.

4. Applications, Use Cases and Examples

This section describes some applications and use cases to illustrate the use of the attributes proposed in this document.

4.1. Managing CGN Port Behavior using RADIUS

In a broadband network, customer information is usually stored on a RADIUS server, and the BNG hosts the NAS. The communication between the NAS and the RADIUS server is triggered by a subscriber when the user signs in to the Internet service, where either PPP or DHCP/DHCPv6 is used. When a user signs in, the NAS sends a RADIUS Access-Request message to the RADIUS server. The RADIUS server validates the request, and if the validation succeeds, it in turn sends back a RADIUS Access-Accept message. The Access-Accept message carries configuration information specific to that user, back to the NAS,

where some of the information would pass on to the requesting user via PPP or DHCP/DHCPv6.

A CGN function in a broadband network would most likely reside on a BNG. In that case, parameters for CGN port/identifier mapping behavior for users can be configured on the RADIUS server. When a user signs in to the Internet service, the associated parameters can be conveyed to the NAS, and proper configuration is accomplished on the CGN device for that user.

Also, CGN operation status such as CGN port/identifier allocation and de-allocation for a specific user on the BNG can also be transmitted back to the RADIUS server for accounting purpose using the RADIUS protocol.

RADIUS protocol has already been widely deployed in broadband networks to manage BNG, thus the functionality described in this specification introduces little overhead to the existing network operation.

In the following sub-sections, we describe how to manage CGN behavior using RADIUS protocol, with required RADIUS extensions proposed in Section 3.

4.1.1.1. Configure IP Port Limit for a User

In the face of IPv4 address shortage, there are currently proposals to multiplex multiple subscribers' connections over a smaller number of shared IPv4 addresses, such as Carrier Grade NAT [RFC6888], Dual-Stack Lite [RFC6333], NAT64 [RFC6146], etc. As a result, a single IPv4 public address may be shared by hundreds or even thousands of subscribers. As indicated in [RFC6269], it is therefore necessary to impose limits on the total number of ports available to an individual subscriber to ensure that the shared resource, i.e., the IPv4 address remains available in some capacity to all the subscribers using it, and port limiting is also documented in [RFC6888] as a requirement.

The IP port limit imposed to a specific subscriber may be on the total number of TCP and UDP ports plus the number of ICMP identifiers, or with other granularities as defined in Section 3.1.2.

The per-subscriber based IP port limit is configured on a RADIUS server, along with other user information such as credentials. The value of these IP port limit is based on service agreement and its specification is out of the scope of this document.

When a subscriber signs in to the Internet service successfully, the IP port limit for the subscriber is passed to the BNG based NAS,

where CGN also locates, using a new RADIUS attribute called IP-Port-Limit (defined in Section 3.1.2), along with other configuration parameters. While some parameters are passed to the subscriber, the IP port limit is recorded on the CGN device for imposing the usage of TCP/UDP ports and ICMP identifiers for that subscriber.

Figure 14 illustrates how RADIUS protocol is used to configure the maximum number of TCP/UDP ports for a given subscriber on a NAT44 device.

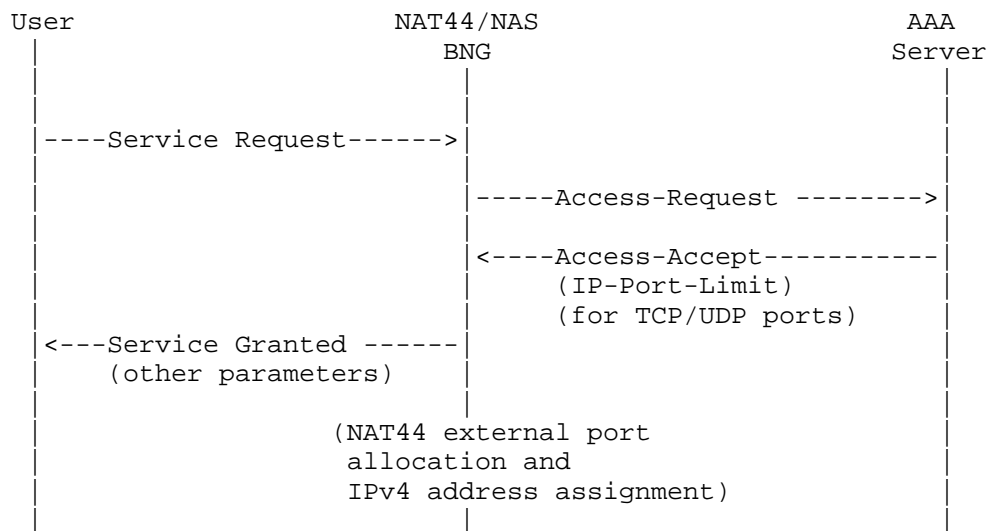


Figure 14: RADIUS Message Flow for Configuring NAT44 Port Limit

The IP port limit created on a CGN device for a specific user using RADIUS extension may be changed using RADIUS CoA message [RFC5176] that carries the same RADIUS attribute. The CoA message may be sent from the RADIUS server directly to the NAS, which once accepts and sends back a RADIUS CoA ACK message, the new IP port limit replaces the previous one.

Figure 15 illustrates how RADIUS protocol is used to increase the TCP/UDP port limit from 1024 to 2048 on a NAT44 device for a specific user.

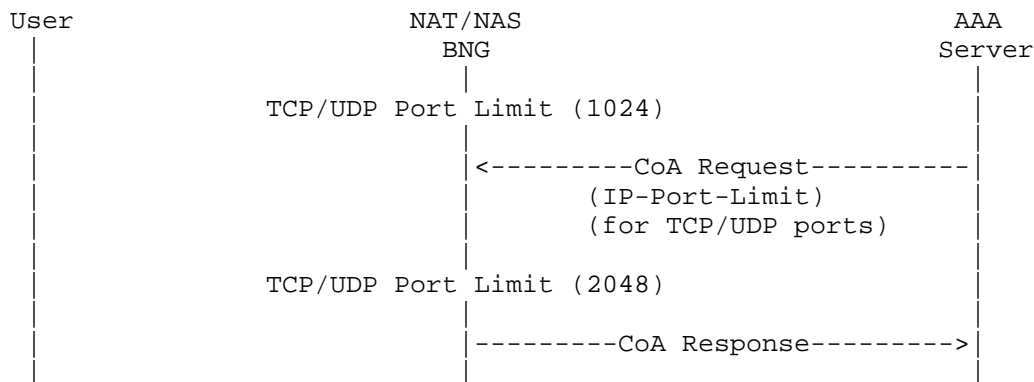


Figure 15: RADIUS Message Flow for changing a user's NAT44 port limit

4.1.1.2. Report IP Port Allocation/De-allocation

Upon obtaining the IP port limit for a subscriber, the CGN device needs to allocate a TCP/UDP port or an ICMP identifiers for the subscriber when receiving a new IP flow sent from that subscriber.

As one practice, a CGN may allocate a bulk of TCP/UDP ports or ICMP identifiers once at a time for a specific user, instead of one port/identifier at a time, and within each port bulk, the ports/identifiers may be randomly distributed or in consecutive fashion. When a CGN device allocates bulk of TCP/UDP ports and ICMP identifiers, the information can be easily conveyed to the RADIUS server by a new RADIUS attribute called the IP-Port-Range (defined in Section 3.1.3). The CGN device may allocate one or more TCP/UDP port ranges or ICMP identifier ranges, or generally called IP port ranges, where each range contains a set of numbers representing TCP/UDP ports or ICMP identifiers, and the total number of ports/identifiers must be less or equal to the associated IP port limit imposed for that subscriber. A CGN device may choose to allocate a small port range, and allocate more at a later time as needed; such practice is good because its randomization in nature.

At the same time, the CGN device also needs to decide the shared IPv4 address for that subscriber. The shared IPv4 address and the pre-allocated IP port range are both passed to the RADIUS server.

When a subscriber initiates an IP flow, the CGN device randomly selects a TCP/UDP port or ICMP identifier from the associated and pre-allocated IP port range for that subscriber to replace the original source TCP/UDP port or ICMP identifier, along with the replacement of the source IP address by the shared IPv4 address.

A CGN device may decide to "free" a previously assigned set of TCP/UDP ports or ICMP identifiers that have been allocated for a specific subscriber but not currently in use, and with that, the CGN device must send the information of the de-allocated IP port range along with the shared IPv4 address to the RADIUS server.

Figure 16 illustrates how RADIUS protocol is used to report a set of ports allocated and de-allocated, respectively, by a NAT44 device for a specific user to the RADIUS server.

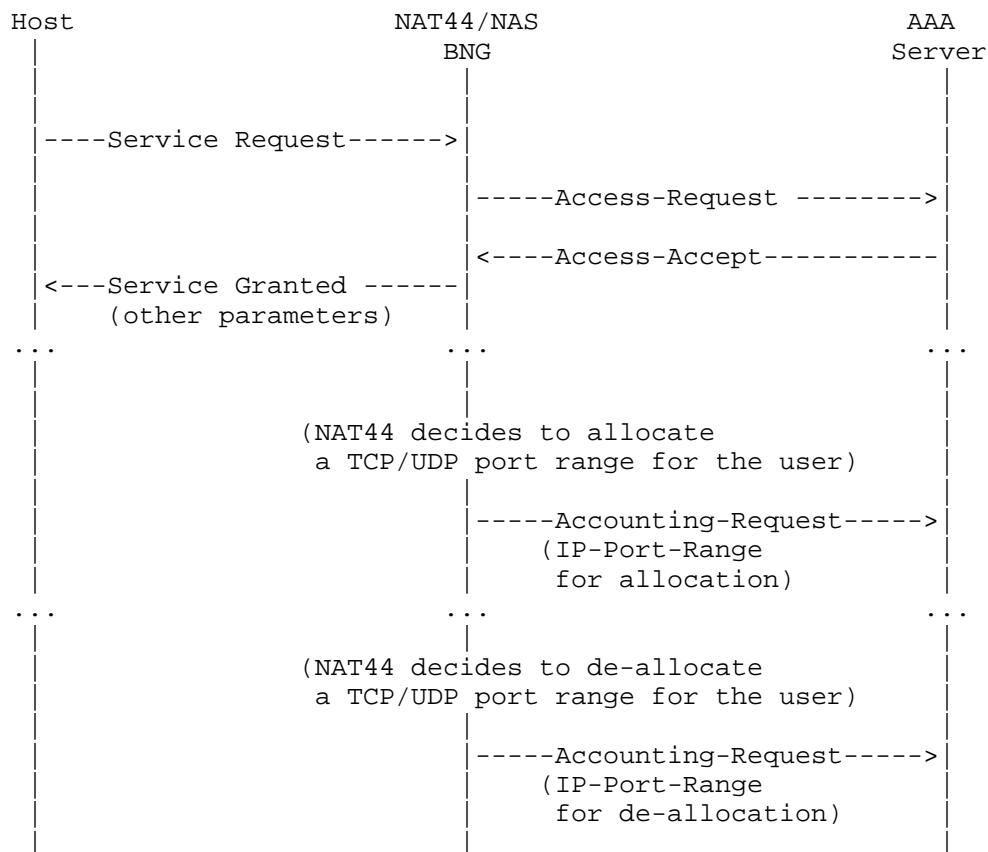


Figure 16: RADIUS Message Flow for reporting NAT44 allocation/de-allocation of a port set

4.1.3. Configure Forwarding Port Mapping

In most scenarios, the port mapping on a NAT device is dynamically created when the IP packets of an IP connection initiated by a user arrives. For some applications, the port mapping needs to be pre-

defined allowing IP packets of applications from outside a CGN device to pass through and "port forwarded" to the correct user located behind the CGN device.

Port Control Protocol [RFC6887], provides a mechanism to create a mapping from an external IP address and port to an internal IP address and port on a CGN device just to achieve the "port forwarding" purpose. PCP is a server-client protocol capable of creating or deleting a mapping along with a rich set of features on a CGN device in dynamic fashion. In some deployment, all users need is a few, typically just one pre-configured port mapping for applications such as web cam at home, and the lifetime of such a port mapping remains valid throughout the duration of the customer's Internet service connection time. In such an environment, it is possible to statically configure a port mapping on the RADIUS server for a user and let the RADIUS protocol to propagate the information to the associated CGN device.

Figure 17 illustrates how RADIUS protocol is used to configure a forwarding port mapping on a NAT44 device by using RADIUS protocol.

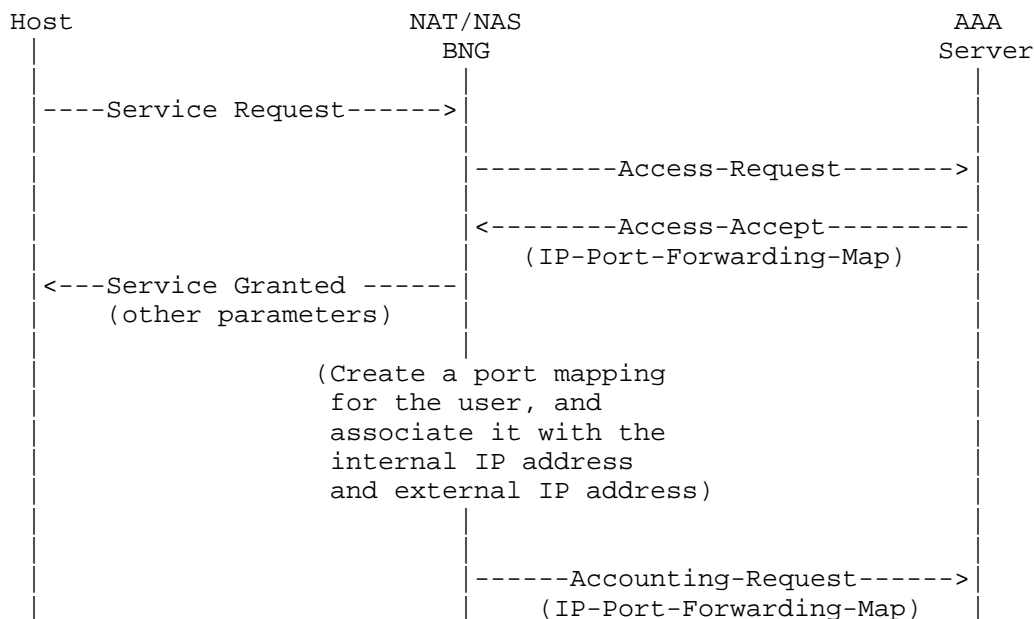


Figure 17: RADIUS Message Flow for configuring a forwarding port mapping

A port forwarding mapping that is created on a CGN device using RADIUS extension as described above may also be changed using RADIUS

CoA message [RFC5176] that carries the same RADIUS associate. The CoA message may be sent from the RADIUS server directly to the NAS, which once accepts and sends back a RADIUS CoA ACK message, the new port forwarding mapping then replaces the previous one.

Figure 18 illustrates how RADIUS protocol is used to change an existing port mapping from (a:X) to (a:Y), where "a" is an internal port, and "X" and "Y" are external ports, respectively, for a specific user with a specific IP address

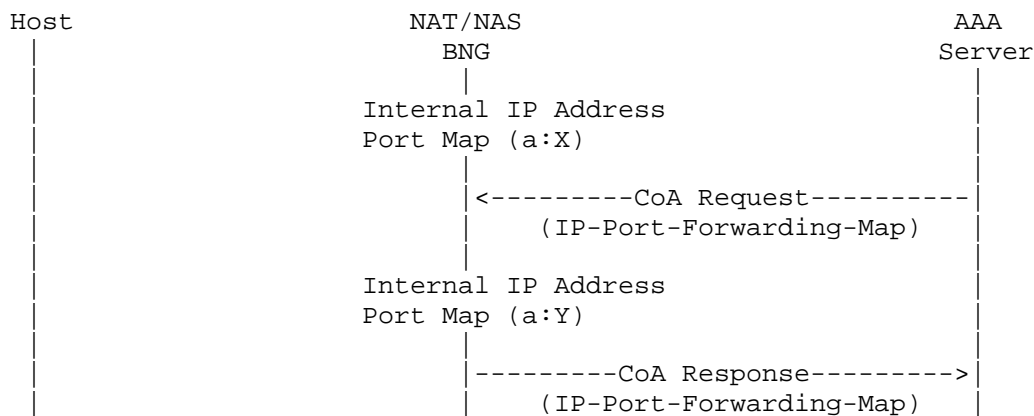


Figure 18: RADIUS Message Flow for changing a user's forwarding port mapping

4.1.4. An Example

An Internet Service Provider (ISP) assigns TCP/UDP 500 ports for the subscriber Joe. This number is the limit that can be used for TCP/UDP ports on a NAT44 device for Joe, and is configured on a RADIUS server. Also, Joe asks for a pre-defined port forwarding mapping on the NAT44 device for his web cam applications (external port 5000 maps to internal port 80).

When Joe successfully connects to the Internet service, the RADIUS server conveys the TCP/UDP port limit (1000) and the forwarding port mapping (external port 5000 to internal port 80) to the NAT44 device, using IP-Port-Limit attribute and IP-Port-Forwarding-Map attribute, respectively, carried by an Access-Accept message to the BNG where NAS and CGN co-located.

Upon receiving the first outbound IP packet sent from Joe's laptop, the NAT44 device decides to allocate a small port pool that contains 40 consecutive ports, from 3500 to 3540, inclusively, and also assign a shared IPv4 address 192.0.2.15, for Joe. The NAT44 device also

randomly selects one port from the allocated range (say 3519) and use that port to replace the original source port in outbound IP packets.

For accounting purpose, the NAT44 device passes this port range (3500-3540) and the shared IPv4 address 192.0.2.15 together to the RADIUS server using IP-Port-Range attribute carried by an Accounting-Request message.

When Joe works on more applications with more outbound IP sessions and the port pool (3500-3540) is close to exhaust, the NAT44 device allocates a second port pool (8500-8800) in a similar fashion, and also passes the new port range (8500-8800) and IPv4 address 192.0.2.15 together to the RADIUS server using IP-Port-Range attribute carried by an Accounting-Request message. Note when the CGN allocates more ports, it needs to assure that the total number of ports allocated for Joe is within the limit.

Joe decides to upgrade his service agreement with more TCP/UDP ports allowed (up to 1000 ports). The ISP updates the information in Joe's profile on the RADIUS server, which then sends a CoA-Request message that carries the IP-Port-Limit attribute with 1000 ports to the NAT44 device; the NAT44 device in turn sends back a CoA-ACK message. With that, Joe enjoys more available TCP/UDP ports for his applications.

When Joe travels, most of the IP sessions are closed with their associated TCP/UDP ports released on the NAT44 device, which then sends the relevant information back to the RADIUS server using IP-Port-Range attribute carried by Accounting-Request message.

Throughout Joe's connection with his ISP Internet service, applications can communicate with his web cam at home from external realm directly traversing the pre-configured mapping on the CGN device.

When Joe disconnects from his Internet service, the CGN device will de-allocate all TCP/UDP ports as well as the port-forwarding mapping, and send the relevant information to the RADIUS server.

4.2. Report Assigned Port Set for a Visiting UE

Figure 19 illustrates an example of the flow exchange which occurs when a visiting UE connects to a CPE offering WLAN service.

For identification purposes (see [RFC6967]), once the CPE assigns a port set, it issues a RADIUS message to report the assigned port set.

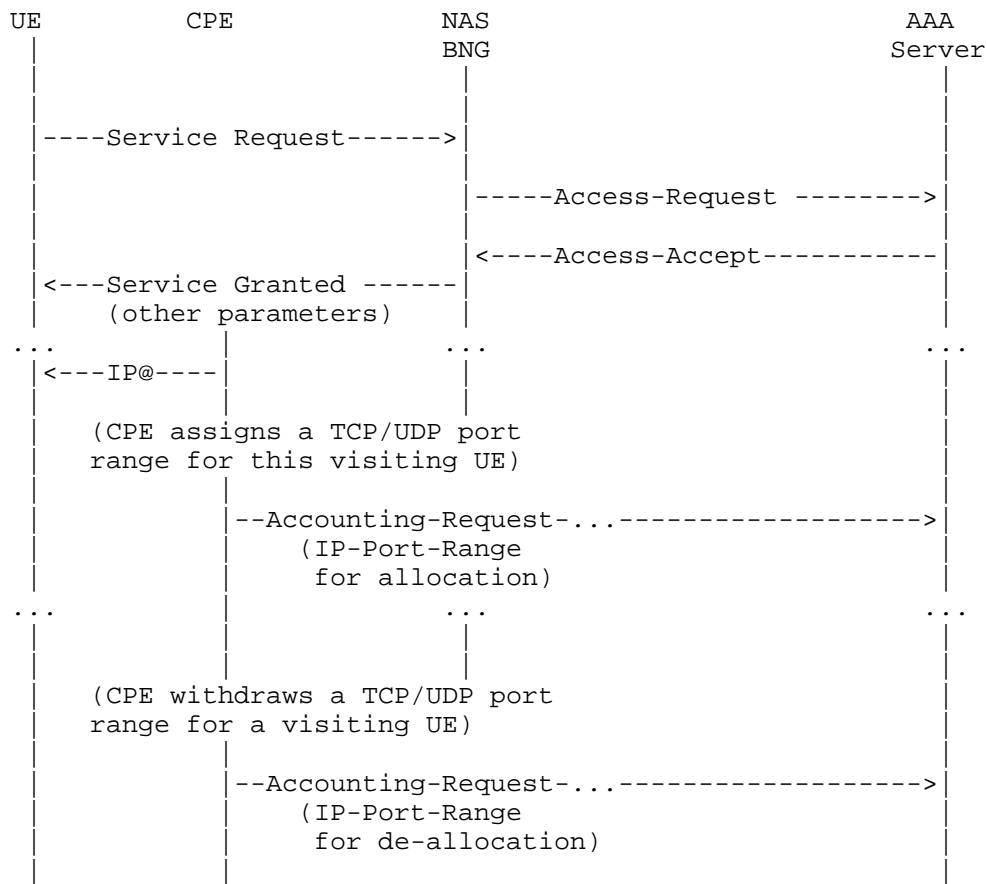


Figure 19: RADIUS Message Flow for reporting CPE allocation/de-allocation of a port set to a visiting UE

5. Table of Attributes

This document proposes three new RADIUS attributes and their formats are as follows:

- o IP-Port-Limit: TBA1.TBA2.{TBA2-1..TBA2-5}.[TBA3, {TBA4}]
- o IP-Port-Range: TBA1.TBA2.{TBA2-1..TBA2-5}.[TBA8, TBA9, TBA10, {TBA4}, {TBA11}].
- o IP-Port-Forwarding-Map: TBA1.TBA2.{TBA2-1 .. TBA2-5}.[TBA6, TBA7, TBA5, {TBA4}]

The following table provides a guide as what type of RADIUS packets that may contain these attributes, and in what quantity.

Request	Accept	Reject	Challenge	Acct. Request	#	Attribute
0+	0+	0	0	0+	TBA	IP-Port-Limit
0	0	0	0	0+	TBA	IP-Port-Range
0+	0+	0	0	0+	TBA	IP-Port-Forwarding-Map

The following table defines the meaning of the above table entries.

0 This attribute MUST NOT be present in packet.

0+ Zero or more instances of this attribute MAY be present in packet.

6. Security Considerations

This document does not introduce any security issue than what has been identified in [RFC2865].

7. IANA Considerations

This document requires new code point assignments for the new RADIUS attributes as follows:

- o TBA1 (refer to Section 3.1.1): This value is for the Radius Type field and should be allocated from the number space of Extended-Type-1 (241), Extended-Type-2 (242), Extended-Type-3 (243), or Extended-Type-4 (244) per [RFC6929].
- o TBA2 (refer to Section 3.1.1): This value is for the Extended-Type field and should be allocated from the Short Extended Space per [RFC6929].
- o TBA2-1, TBA2-2, TBA2-3, TBA2-4, and TBA2-5 (refer to Section 3.1.1): These values are for the Type field of IP-Port-Type TLV that is within the TBA2 container, and they should be allocated as TLV data type and effectively extend the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.
- o TBA3 (refer to Section 3.1.2): This value is for the type field of IP-Port-Limit TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.TBA3.
- o TBA4 (refer to Section 3.2.2): This value is for the Type field of IP-Port-Ext-IPv4-Addr TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.[TBA4...].

- o TBA5 (refer to Section 3.2.3): This value is for the Type field of IP-Port-Int-IP-Addr TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.[TBA5...].
- o TBA6 (refer to Section 3.2.4): This value is for the Type field of IP-Port-Int-Port TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.[TBA6...].
- o TBA7 (refer to Section 3.2.5): This value is for the Type field of IP-Port-Ext-port TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.[TBA7...].
- o TBA8 (refer to Section 3.2.6): This value is for the Type field of IP-Port-Alloc TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.[TBA8...].
- o TBA9 (refer to Section 3.2.7): This value is for the Type field of IP-Port-Range-Start TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.[TBA9...].
- o TBA10 (refer to Section 3.2.8): This value is for the Type field of IP-Port-Range-End TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.[TBA10...].
- o TBA11 (refer to Section 3.2.9): This value is for the Type field of IP-Port-Local-Id TLV. It should be allocated as TLV data type and it extends the attribute tree as TBA1.TBA2.{TBA2-1, TBA2-2, TBA2-3, TBA2-4, TBA2-5}.[TBA11...].

8. Acknowledgements

Many thanks to Dan Wing, Roberta Maglione, Daniel Derksen, David Thaler, Alan Dekok, Lionel Morand, and Peter Deacon for their useful comments and suggestions.

9. References

9.1. Normative References

- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.
- [RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

9.2. Informative References

- [I-D.gundavelli-v6ops-community-wifi-svcs]
Gundavelli, S., Grayson, M., Seite, P., and Y. Lee, "Service Provider Wi-Fi Services Over Residential Architectures", draft-gundavelli-v6ops-community-wifi-svcs-06 (work in progress), April 2013.
- [I-D.ietf-software-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-software-lw4over6-10 (work in progress), June 2014.
- [I-D.miles-behave-l2nat]
Miles, D. and M. Townsley, "Layer2-Aware NAT", draft-miles-behave-l2nat-00 (work in progress), March 2009.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.
- [RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

- [RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.
- [RFC6619] Arkko, J., Eggert, L., and M. Townsley, "Scalable Operation of Address Translators with Per-Interface Bindings", RFC 6619, June 2012.
- [RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.
- [RFC6888] Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, April 2013.
- [RFC6967] Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Potential Solutions for Revealing a Host Identifier (HOST_ID) in Shared Address Deployments", RFC 6967, June 2013.

Authors' Addresses

Dean Cheng
Huawei
2330 Central Expressway
Santa Clara, California 95050
USA

Email: dean.cheng@huawei.com

Jouni Korhonen
Broadcom
Porkkalankatu 24
FIN-00180 Helsinki
Finland

Email: jouni.nospam@gmail.com

Mohamed Boucadair
France Telecom
Rennes
France

Email: mohamed.boucadair@orange.com

Senthil Sivakumar
Cisco Systems
7100-8 Kit Creek Road
Research Triangle Park, North Carolina
USA

Email: ssenthil@cisco.com

RADEXT Working Group
INTERNET-DRAFT
Obsoletes: 4282
Category: Standards Track
<draft-ietf-radext-nai-06.txt>
17 June 2014

DeKok, Alan
FreeRADIUS

The Network Access Identifier
draft-ietf-radext-nai-06

Abstract

In order to provide inter-domain authentication services, it is necessary to have a standardized method that domains can use to identify each other's users. This document defines the syntax for the Network Access Identifier (NAI), the user identity submitted by the client prior to accessing network resources. This document is a revised version of RFC 4282, which addresses issues with international character sets, as well as a number of other corrections to the previous document.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 17, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

Appendix A - Changes from RFC4282	3
1. Introduction	4
1.1. Terminology	5
1.2. Requirements Language	6
1.3. Purpose	7
1.4. Motivation	7
2. NAI Definition	8
2.1. UTF-8 Syntax and Normalization	8
2.2. Formal Syntax	9
2.3. NAI Length Considerations	10
2.4. Support for Username Privacy	11
2.5. International Character Sets	11
2.6. The Normalization Process	12
2.6.1. Issues with the Normalization Process	13
2.7. Use in Other Protocols	14
3. Routing inside of AAA Systems	15
3.1. Compatibility with Email Usernames	16
3.2. Compatibility with DNS	16
3.3. Realm Construction	17
3.3.1. Historical Practices	18
3.4. Examples	18
4. Security Considerations	19
5. Administration of Names	20
6. IANA Considerations	20
7. References	20
7.1. Normative References	20
7.2. Informative References	21
Appendix A - Changes from RFC4282	23

1. Introduction

Considerable interest exists for a set of features that fit within the general category of inter-domain authentication, or "roaming capability" for network access, including dialup Internet users, Virtual Private Network (VPN) usage, wireless LAN authentication, and other applications. Interested parties have included the following:

- * Regional Internet Service Providers (ISPs) operating within a particular state or province, looking to combine their efforts with those of other regional providers to offer dialup service over a wider area.
- * National ISPs wishing to combine their operations with those of one or more ISPs in another nation to offer more comprehensive dialup service in a group of countries or on a continent.
- * Wireless LAN hotspots providing service to one or more ISPs.
- * Businesses desiring to offer their employees a comprehensive package of dialup services on a global basis. Those services may include Internet access as well as secure access to corporate intranets via a VPN, enabled by tunneling protocols such as the Point-to-Point Tunneling Protocol (PPTP) [RFC2637], the Layer 2 Forwarding (L2F) protocol [RFC2341], the Layer 2 Tunneling Protocol (L2TP) [RFC2661], and the IPsec tunnel mode [RFC4301].
- * Other protocols which are interested in leveraging the users credentials in order to take advantage of an existing authentication framework.

In order to enhance the interoperability of these services, it is necessary to have a standardized method for identifying users. This document defines syntax for the Network Access Identifier (NAI). Examples of implementations that use the NAI, and descriptions of its semantics, can be found in [RFC2194].

When the NAI was defined for network access, it had the side effect of defining an identifier which could be used in non-AAA systems. Some systems defined identifiers which were compatible with the NAI, and deployments used the NAI. This process simplified the management of credentials, by re-using the same credential in multiple situations. We suggest that this re-use is good practice. The alternative is to have protocol-specific identifiers, which increases cost to both user and administrator.

The goal of this document is to define the format of an identifier which can be used in many protocols. A protocol may transport an

encoded version of the NAI (e.g. '.' as %2E). However, the definition of the NAI is protocol independent. We hope to encourage the wide-spread adoption of the NAI as an identifier. This adoption will decrease work required to leverage identification and authentication in other protocols. It will also decrease the complexity of systems for end users and administrators.

We note that this document only suggest that the NAI be used, but does not require it. Many protocols already define their own identifier formats. Some of these are incompatible with the NAI, while others allow the NAI in addition to non-NAI identifiers. This definition of the NAI has no requirements on protocol specifications, implementations, or deployments. We suggest that using a standard identifier format is preferable to using multiple incompatible identifier formats.

This document is a revised version of [RFC4282], which originally defined internationalized NAIs. Differences and enhancements compared to that document are listed in Appendix A.

1.1. Terminology

This document frequently uses the following terms:

"Local" or "localized" text

Text which is either in non-UTF-8, or in non-normalized form. The character set, encoding, and locale are (in general) unknown to Authentication, Authorization, and Accounting (AAA) network protocols. The client which "knows" the locale may have a different concept of this text than other AAA entities, which do not know the same locale.

Network Access Identifier

The Network Access Identifier (NAI) is the user identity submitted by the client during network access authentication. The purpose of the NAI is to identify the user as well as to assist in the routing of the authentication request. Please note that the NAI may not necessarily be the same as the user's email address or the user identity submitted in an application layer authentication.

Network Access Server

The Network Access Server (NAS) is the device that clients connect to in order to get access to the network. In PPTP terminology, this is referred to as the PPTP Access Concentrator (PAC), and in L2TP terminology, it is referred to as the L2TP Access

Concentrator (LAC). In IEEE 802.11, it is referred to as an Access Point.

Roaming Capability

Roaming capability can be loosely defined as the ability to use any one of multiple Internet Service Providers (ISPs), while maintaining a formal, customer-vendor relationship with only one. Examples of cases where roaming capability might be required include ISP "confederations" and ISP-provided corporate network access support.

Normalization Canonicalization

These terms are defined in [RFC6365] Section 4. We incorporate the definitions here by reference.

Locale

This term is defined in [RFC6365] Section 8. We incorporate the definition here by reference.

Tunneling Service

A tunneling service is any network service enabled by tunneling protocols such as PPTP, L2F, L2TP, and IPsec tunnel mode. One example of a tunneling service is secure access to corporate intranets via a Virtual Private Network (VPN).

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.3. Purpose

As described in [RFC2194], there are a number of providers offering network access services, and the number of Internet Service Providers involved in roaming consortia is increasing rapidly.

In order to be able to offer roaming capability, one of the requirements is to be able to identify the user's home authentication server. For use in roaming, this function is accomplished via the Network Access Identifier (NAI) submitted by the user to the NAS in the initial network authentication. It is also expected that NASes will use the NAI as part of the process of opening a new tunnel, in order to determine the tunnel endpoint.

We also hope that other protocols can take advantage of the NAI. Many protocols include authentication capabilities, including defining their own identifier formats. These identifiers can then end up being transported in AAA protocols, when those systems want to leverage AAA for user authentication. There is therefore a need for a definition of a user identifier which can be used in multiple protocols.

While we define the NAI here, we recognize that existing protocols and deployments do not always use it. AAA systems **MUST** therefore be able to handle user identifiers which are not in the NAI format. The process by which that is done is outside of the scope of this document.

We note that this document does not make any protocol-specific definitions for an identifier format, and it does not make changes to any existing protocol. Instead, it defines a protocol-independent form for the NAI. It is hoped that the NAI is a user identifier which can be used in multiple protocols.

Using a common identifier simplifies deployments, as there is no need to maintain multiple identifiers for one user. It simplifies protocols requiring authentication, as they no longer need to specify protocol-specific format for user identifiers. It increases security, as it multiple identifiers allow attackers to make contradictory claims without being detected.

In short, having a standard is better than having no standard at all.

1.4. Motivation

The changes from [RFC4282] are listed in detail in Appendix A. However, some additional discussion is appropriate to motivate those changes.

The motivation to revise [RFC4282] began with internationalization concerns raised in the context of [EDUROAM]. Section 2.1 of [RFC4282] defines ABNF for realms which limits the realm grammar to English letters, digits, and the hyphen "-" character. The intent appears to have been to encode, compare, and transport realms with the ToASCII operation defined in [RFC5890]. There are a number of problems with this approach:

- * The [RFC4282] ABNF is not aligned with internationalization of DNS.
- * The requirement in Section 2.1 that realms are ASCII conflicts with the Extensible Authentication Protocol (EAP) and RADIUS, which are both 8-bit clean, and which both recommend the use of UTF-8 for identifiers.
- * Section 2.4 required mappings that are language-specific, and which are nearly impossible for intermediate nodes to perform correctly without information about that language.
- * Section 2.4 requires normalization of user names, which may conflict with local system or administrative requirements.
- * The recommendations in Section 2.4 for treatment of bidirectional characters have proven to be unworkable.
- * The prohibition against use of unassigned code points in Section 2.4 effectively prohibits support for new scripts.
- * No Authentication, Authorization, and Accounting (AAA) client, proxy, or server has implemented any of the requirements in [RFC4282] Section 2.4, among other sections.

With international roaming growing in popularity, it is important for these issues to be corrected in order to provide robust and interoperable network services.

2. NAI Definition

2.1. UTF-8 Syntax and Normalization

UTF-8 characters can be defined in terms of octets using the following ABNF [RFC5234], taken from [RFC3629]:

UTF8-xtra-char = UTF8-2 / UTF8-3 / UTF8-4

UTF8-2 = %xC2-DF UTF8-tail

```
UTF8-3      =  %xE0 %xA0-BF UTF8-tail /
                %xE1-EC 2(UTF8-tail) /
                %xED %x80-9F UTF8-tail /
                %xEE-EF 2(UTF8-tail)

UTF8-4      =  %xF0 %x90-BF 2( UTF8-tail ) /
                %xF1-F3 3( UTF8-tail ) /
                %xF4 %x80-8F 2( UTF8-tail )

UTF8-tail   =  %x80-BF
```

These are normatively defined in [RFC3629], but are repeated in this document for reasons of convenience.

See [RFC5198] and section 2.6 of this specification for a discussion of normalization. Strings which are not in Normal Form Composed (NFC) are not valid NAIs and SHOULD NOT be treated as such. Implementations which expect to receive a NAI, but which instead receive non-normalised (but otherwise valid) UTF-8 strings instead SHOULD attempt to create a local version of the NAI, which is normalized from the input identifier. This local version can then be used for local processing.

Systems MAY accept user identifiers in forms other than the NAI. This specification does not forbid that practice. It only codifies the format and interpretation of the NAI. We cannot expect to change existing protocols or practices. We can, however, suggest that using a consistent form for a user identifier is of a benefit to the community.

Where protocols carry identifiers which are expected to be transported over an AAA protocol, it is RECOMMENDED that the identifiers be in NAI format. Where the identifiers are not in the NAI format, it is up to the AAA systems to discover this, and to process them. This document does not suggest how that is done. However, existing practice indicates that it is possible.

We expect that with wider use of internationalized domain names, existing practices will be inadequate. We therefore define the NAI, which is a user identifier that can correctly deal with internationalized identifiers.

2.2. Formal Syntax

The grammar for the NAI is given below, described in Augmented Backus-Naur Form (ABNF) as documented in [RFC5234].

```
nai          =  utf8-username
```

```

nai           =/  "@" utf8-realm
nai           =/  utf8-username "@" utf8-realm

utf8-username = dot-string
dot-string    = string
dot-string    =/ dot-string "." string
string        = utf8-atext
string        =/ string utf8-atext

utf8-atext    = ALPHA / DIGIT /
               "!" / "#" /
               "$" / "%" /
               "&" / "'" /
               "*" / "+" /
               "-" / "/" /
               "=" / "?" /
               "^" / "_" /
               "`" / "{" /
               "|" / "}" /
               "~" /
               UTF8-xtra-char

utf8-realm    = 1*( label "." ) label

label         = utf8-rtext *(ldh-str)
ldh-str       = *( utf8-rtext / "-" ) utf8-rtext
utf8-rtext    = ALPHA / DIGIT / UTF8-xtra-char

```

2.3. NAI Length Considerations

Devices handling NAIs MUST support an NAI length of at least 72 octets. Devices SHOULD support an NAI length of 253 octets. However, the following implementation issues should be considered:

- * NAI octet length constraints may impose a more severe constraint on the number of UTF-8 characters.
- * NAIs are often transported in the User-Name attribute of the Remote Authentication Dial-In User Service (RADIUS) protocol. Unfortunately, RFC 2865 [RFC2865], Section 5.1, states that "the ability to handle at least 63 octets is recommended." As a result, it may not be possible to transfer NAIs beyond 63 octets through all devices. In addition, since only a single User-Name attribute may be included in a RADIUS message and the maximum attribute length is 253 octets; RADIUS is unable to support NAI lengths beyond 253 octets.
- * NAIs can also be transported in the User-Name attribute of

Diameter [RFC6733], which supports content lengths up to $2^{24} - 9$ octets. As a result, NAIs processed only by Diameter nodes can be very long. However, an NAI transported over Diameter may eventually be translated to RADIUS, in which case the above limitations will apply.

* NAIs may be transported in other protocols. Each protocol can have its own limitations on maximum NAI length. The above criteria should permit the widest use, and widest possible inter-operability of the NAI.

2.4. Support for Username Privacy

Interpretation of the username part of the NAI depends on the realm in question. Therefore, the utf8-username portion SHOULD be treated as opaque data when processed by nodes that are not a part of the authoritative domain (in the sense of Section 4) for that realm.

In some situations, NAIs are used together with a separate authentication method that can transfer the username part in a more secure manner to increase privacy. In this case, NAIs MAY be provided in an abbreviated form by omitting the username part. Omitting the username part is RECOMMENDED over using a fixed username part, such as "anonymous", since it provides an unambiguous way to determine whether the username is intended to uniquely identify a single user. However, current practice is to use the username "anonymous" instead of omitting the username part. This behavior is also permitted.

For roaming purposes, it is typically necessary to locate the appropriate backend authentication server for the given NAI before the authentication conversation can proceed. As a result, the realm portion is typically required in order for the authentication exchange to be routed to the appropriate server.

2.5. International Character Sets

This specification allows both international usernames and realms. International usernames are based on the use of Unicode characters, encoded as UTF-8. Internationalization of the realm portion of the NAI is based on "Internationalized Email Headers" [RFC5335].

In order to ensure a canonical representation, characters of the username portion in an NAI MUST match the ABNF in this specification as well as the requirements specified in [RFC5891]. In practice, these requirements consist of the following item:

* Realms MUST be of the form that can be registered as a

Fully Qualified Domain Name (FQDN) within the DNS.

This list is significantly shorter and simpler than the list in Section 2.4 of [RFC4282]. The form suggested in [RFC4282] depended on intermediate nodes performing canonicalizations based on insufficient information, which meant that the form was not canonical.

Specifying the realm requirement as above means that the requirements depend on specifications that are referenced here, rather than copied here. This allows the realm definition to be updated when the referenced documents change, without requiring a revision of this specification.

One caveat on the above recommendation is the issues noted in [RFC6912]. That document notes that there are additional restrictions around DNS registration which forbid some code points from being valid in a DNS U-label. These restrictions cannot be expressed algorithmically.

For this specification, that caveat means the following. Realms not matching the above ABNF are not valid NAIs. However, some realms which do match the ABNF are still invalid NAIs. That is, matching the ABNF is a necessary, but not sufficient, requirement for an NAI.

In general, the above requirement means following the requirements specified in [RFC5891].

2.6. The Normalization Process

Conversion to Unicode as well as normalization SHOULD be performed by end systems that take "local" text as input. These systems are best suited to determine the users intent, and can best convert from "local" text to a normalized form.

Other AAA systems such as proxies do not have access to locale and character set information that is available to end systems. Therefore, they can not always convert local input to Unicode.

That is, all processing of NAIs from "local" character sets and locales to UTF-8 SHOULD be performed by edge systems, prior to the NAIs entering the AAA system. Inside of an AAA system, NAIs are sent over the wire in their canonical form, and this canonical form is used for all NAI and/or realm comparisons.

Copying of localized text into fields that can subsequently be placed into the RADIUS User-Name attribute is problematic. This practice can result in a AAA proxy encountering non-UTF8 characters within

what it expects to be an NAI. An example of this requirement is [RFC3579] Section 2.1, which states:

the NAS MUST copy the contents of the Type-Data field of the EAP-Response/Identity received from the peer into the User-Name attribute

As a result, AAA proxies expect the contents of the EAP-Response/Identity sent by an EAP supplicant to consist of UTF-8 characters, not localized text. Using localized text in AAA username or identity fields means that realm routing becomes difficult or impossible.

In contrast to [RFC4282] Section 2.4, we expect AAA systems to perform NAI comparisons, matching, and AAA routing based on the NAI as it is received. This specification provides a canonical representation, ensures that intermediate systems such as AAA proxies do not need to perform translations, and can be expected to work through systems that are unaware of international character sets.

In short,

- * End systems using "localized" text SHOULD normalize the NAI prior to it being used as an identifier in an authentication protocol.
- * AAA systems SHOULD NOT normalize the NAI, as they may not have sufficient information to perform the normalization.

For example, much of the common realm routing can be done on the "utf8-realm" portion of NAI, through simple checks for equality. This routing can be done even if the AAA proxy is unaware of internationalized domain names. All that is required is for the AAA proxy to be able to enter, store, and compare 8-bit data.

2.6.1. Issues with the Normalization Process

We recognize that the requirements in the preceding section are not implemented today. For example, most EAP implementations use a user identifier which is passed to them from some other local system. This identifier is treated as an opaque blob, and is placed as-is into the EAP Identity field. Any subsequent system which receives that identifier is assumed to be able to understand and process it.

This opaque blob unfortunately can contain localized text, which means that the AAA systems have to process that text.

These limitations have the following theoretical and practical

implications.

- * "local" systems used today generally do not normalize the NAI
- * Therefore AAA systems SHOULD attempt to normalize the NAI

The suggestion in the above sentence contradicts the suggestion in the previous section. This is the reality of imperfect protocols.

Where the user identifier can be normalized, or determined to be in normal form, the normal form MUST be used as the NAI. In all other circumstances, the user identifier MUST NOT be treated as an NAI. That data is still, however, a user identifier. AAA systems MUST NOT fail authentication simply because the user identifier is not an NAI.

That is, when the realm portion of the NAI is not recognized by an AAA server, it SHOULD try to normalize the NAI into NFC form. That normalized form can then be used to see if the realm matches a known realm. If no match is found, the original form of the NAI SHOULD be used in all subsequent processing.

The AAA server may also convert realms to punycode, and perform all realm comparisons on the resulting punycode strings. This conversion follows the recommendations above, but may have different operational effects and failure modes.

2.7. Use in Other Protocols

As noted earlier, the NAI MAY be used in other, non-AAA protocols. It is RECOMMENDED that the definition given here be used unchanged. Using other definitions for user identifiers may hinder interoperability, along with the users ability to authenticate successfully. It is RECOMMENDED that protocols requiring the use of a user identifier reference this specification, and suggest that the use of an NAI is RECOMMENDED.

We cannot require other protocols to use the NAI for user identifiers. Their needs are unknown, and unknowable. We simply suggest that interoperability and inter-domain authentication is useful, and should be encouraged.

Where a protocol is 8-bit clean, it can likely transport the NAI as-is, without further modification.

Where a protocol is not 8-bit clean, it cannot transport the NAI as-is. Instead, we presume that a protocol-specific transport layer takes care of encoding the NAI on input to the protocol, and decoding it when the NAI exits the protocol. The encoded or escaped version

of the NAI is not a valid NAI, and MUST NOT be presented to the AAA system.

For example, HTTP carries user identifiers, but escapes the '.' character as "%2E" (among others). When we desire HTTP to transport the NAI "fred@example.com", the data as transported will be in the form "fred@example%2Ecom". That data exists only within HTTP, and has no relevance to any AAA system.

Any comparison, validation, or use of the NAI MUST be done on its un-escaped (i.e. utf8-clean) form.

3. Routing inside of AAA Systems

Many AAA systems use the "utf8-realm" portion of the NAI to route requests within a AAA proxy network. The semantics of this operation involves a logical AAA routing table, where the "utf8-realm" portion acts as a key, and the values stored in the table are one or more "next hop" AAA servers.

Intermediate nodes MUST use the "utf8-realm" portion of the NAI without modification to perform this lookup. As noted earlier, intermediate nodes may not have access to the same locale information as the system which injected the NAI into the AAA routing systems. Therefore, almost all "case insensitive" comparisons will be wrong. Where the "utf8-realm" is entirely ASCII, current systems sometimes perform case-insensitive matching on realms. This practice MAY be continued, as it has been shown to work in practice.

We also note that many existing systems have user identifiers which are similar in format to the NAI, but which are not compliant with this specification. For example, they may use non-NFC form, or they may have multiple "@" characters in the user identifier. Intermediate nodes SHOULD normalize non-NFC identifiers to NFC, prior to looking up the "utf8-realm" in the logical routing table. Intermediate nodes MUST NOT modify the identifiers that they forward. The data as entered by the user is inviolate.

The "utf8-realm" provisioned in the logical AAA routing table SHOULD be provisioned to the proxy prior to it receiving any AAA traffic. The "utf8-realm" SHOULD be supplied by the "next hop" or "home" system that also supplies the routing information necessary for packets to reach the next hop.

This "next hop" information may be any of, or all of, the following information: IP address; port; RADIUS shared secret; TLS certificate; DNS host name; or instruction to use dynamic DNS discovery (i.e. look up a record in the "utf8-realm" domain). This list is not

exhaustive, and may be extended by future specifications.

It is RECOMMENDED to use the entirety of the "utf8-realm" for the routing decisions. However, systems MAY use a portion of the "utf8-realm" portion, so long as that portion is a valid "utf8-realm", and that portion is handled as above. For example, routing "fred@example.com" to a "com" destination is forbidden, because "com" is not a valid "utf8-realm". However, routing "fred@sales.example.com" to the "example.com" destination is permissible.

Another reason to forbid the use of a single label (e.g. "fred@sales") is that many systems treat a single label as being a local identifier within their realm. That is, a user logging in as "fred@sales" to a domain "example.com", would be treated as if the NAI was instead "fred@sales.example.com". Permitting the use of a single label would mean changing the interpretation and meaning of a single label, which cannot be done.

3.1. Compatibility with Email Usernames

As proposed in this document, the Network Access Identifier is of the form "user@realm". Please note that while the user portion of the NAI is based on the BNF described in [RFC5198], it has been modified for the purposes of Section 2.2. It does not permit quoted text along with "folding" or "non-folding" whitespace that is commonly used in email addresses. As such, the NAI is not necessarily equivalent to usernames used in e-mail.

However, it is a common practice to use email addresses as user identifiers in AAA systems. The ABNF in Section 2.2 is defined to be close to the "utf8-addr-spec" portion of [RFC5335], while still being compatible with [RFC4282].

In contrast to [RFC4282] Section 2.5, we state that the internationalization requirements for NAIs and email addresses are substantially similar. The NAI and email identifiers may be the same, and both need to be entered by the user and/or the operator supplying network access to that user. There is therefore good reason for the internationalization requirements to be similar.

3.2. Compatibility with DNS

The "utf8-realm" portion of the NAI is intended to be compatible with Internationalized Domain Names (IDNs) [RFC5890]. As defined above, the "utf8-realm" portion as transported within an 8-bit clean protocol such as RADIUS and EAP can contain any valid UTF8 character. There is therefore no reason for a NAS to apply the ToAscii function

to the "utf8-realm" portion of an NAI, prior to placing the NAI into a RADIUS User-Name attribute.

The NAI does not make a distinction between A-labels and U-labels, as those are terms specific to DNS. It is instead an IDNA-valid label, as per the first item in Section 2.3.2.1 of [RFC5890]. As noted in that section, the term "IDNA-valid label" encompasses both of the terms A-label and U-label.

When the realm portion of the NAI is used as the basis for name resolution, it may be necessary to convert internationalized realm names to ASCII using the ToASCII operation defined in [RFC5890]. As noted in [RFC6055] Section 2, resolver Application Programming Interfaces (APIs) are not necessarily DNS-specific, so that the ToASCII operation needs to be applied carefully:

Applications which convert an IDN to A-label form before calling (for example) `getaddrinfo()` will result in name resolution failures if the Punycode name is directly used in such protocols. Having libraries or protocols to convert from A-labels to the encoding scheme defined by the protocol (e.g., UTF-8) would require changes to APIs and/or servers, which IDNA was intended to avoid.

As a result, applications SHOULD NOT assume that non-ASCII names are resolvable using the public DNS and blindly convert them to A-labels without knowledge of what protocol will be selected by the name resolution library.

3.3. Realm Construction

The home realm usually appears in the "utf8-realm" portion of the NAI, but in some cases a different realm can be used. This may be useful, for instance, when the home realm is reachable only via intermediate proxies.

Such usage may prevent interoperability unless the parties involved have a mutual agreement that the usage is allowed. In particular, NAIs MUST NOT use a different realm than the home realm unless the sender has explicit knowledge that (a) the specified other realm is available and (b) the other realm supports such usage. The sender may determine the fulfillment of these conditions through a database, dynamic discovery, or other means not specified here. Note that the first condition is affected by roaming, as the availability of the other realm may depend on the user's location or the desired application.

The use of the home realm MUST be the default unless otherwise configured.

3.3.1. Historical Practices

Some systems have historically used NAI modifications with multiple "prefix" and "suffix" decorations to perform explicit routing through multiple proxies inside of a AAA network. This practice is NOT RECOMMENDED for the following reasons:

- * Using explicit routing paths is fragile, and is unresponsive to changes in the network due to servers going up or down, or to changing business relationships.
- * There is no RADIUS routing protocol, meaning that routing paths have to be communicated "out of band" to all intermediate AAA nodes, and also to all end-user systems (supplicants) expecting to obtain network access.
- * Using explicit routing paths requires thousands, if not millions of end-user systems to be updated with new path information when a AAA routing path changes. This adds huge expense for updates that would be better done at only a few AAA systems in the network.
- * Manual updates to RADIUS paths are expensive, time-consuming, and prone to error.
- * Creating compatible formats for the NAI is difficult when locally-defined "prefixes" and "suffixes" conflict with similar practices elsewhere in the network. These conflicts mean that connecting two networks may be impossible in some cases, as there is no way for packets to be routed properly in a way that meets all requirements at all intermediate proxies.
- * Leveraging the DNS name system for realm names establishes a globally unique name space for realms.

In summary, network practices and capabilities have changed significantly since NAIs were first overloaded to define AAA routes through a network. While explicit path routing was once useful, the time has come for better methods to be used.

3.4. Examples

Examples of valid Network Access Identifiers include the following:

```
bob
joe@example.com
fred@foo-9.example.com
jack@3rd.depts.example.com
```

```
fred.smith@example.com
fred_smith@example.com
fred$@example.com
fred=?#&*+~/^smith@example.com
nancy@eng.example.net
eng.example.net!nancy@example.net
eng%nancy@example.net
@privatecorp.example.net
\(user\)@example.net
```

An additional valid NAI is the following, given as a hex string, as this document can only contain ASCII characters.

```
626f 6240 ceb4 cebf ceba ceb9 cebc ceae 2e63 6f6d
```

Examples of invalid Network Access Identifiers include the following:

```
fred@example
fred@example_9.com
fred@example.net@example.net
fred.@example.net
eng:nancy@example.net
eng;nancy@example.net
(user)@example.net
<nancy>@example.net
```

One example given in [RFC4282] is still permitted by the ABNF, but it is NOT RECOMMENDED because of the use of the ToAscii function to create an ASCII encoding from what is now a valid UTF-8 string.

```
alice@xn--tmonesimerkki-bfbb.example.net
```

4. Security Considerations

Since an NAI reveals the home affiliation of a user, it may assist an attacker in further probing the username space. Typically, this problem is of most concern in protocols that transmit the username in clear-text across the Internet, such as in RADIUS, described in [RFC2865] and [RFC2866]. In order to prevent snooping of the username, protocols may use confidentiality services provided by protocols transporting them, such as RADIUS protected by IPsec [RFC3579] or Diameter protected by TLS [RFC6733].

This specification adds the possibility of hiding the username part in the NAI, by omitting it. As discussed in Section 2.4, this is possible only when NAIs are used together with a separate authentication method that can transfer the username in a secure manner. In some cases, application-specific privacy mechanisms have

also been used with NAIs. For instance, some EAP methods apply method-specific pseudonyms in the username part of the NAI [RFC3748]. While neither of these approaches can protect the realm part, their advantage over transport protection is that privacy of the username is protected, even through intermediate nodes such as NASes.

5. Administration of Names

In order to avoid creating any new administrative procedures, administration of the NAI realm namespace piggybacks on the administration of the DNS namespace.

NAI realm names are required to be unique, and the rights to use a given NAI realm for roaming purposes are obtained coincident with acquiring the rights to use a particular Fully Qualified Domain Name (FQDN). Those wishing to use an NAI realm name should first acquire the rights to use the corresponding FQDN. Administrators **MUST NOT** publicly use an NAI realm without first owning the corresponding FQDN. Private use of unowned NAI realms within an administrative domain is allowed, though it is **RECOMMENDED** that example names be used, such as "example.com".

Note that the use of an FQDN as the realm name does not require use of the DNS for location of the authentication server. While Diameter [RFC6733] supports the use of DNS for location of authentication servers, existing RADIUS implementations typically use proxy configuration files in order to locate authentication servers within a domain and perform authentication routing. The implementations described in [RFC2194] did not use DNS for location of the authentication server within a domain. Similarly, existing implementations have not found a need for dynamic routing protocols or propagation of global routing information. Note also that there is no requirement that the NAI represent a valid email address.

6. IANA Considerations

This document has no actions for IANA.

7. References

7.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March, 1997.

[RFC3629]

Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC5198]

Klensin J., and Padlipsky M., "Unicode Format for Network Interchange", RFC 5198, March 2008

[RFC5234]

Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, January 2008.

[RFC5890]

Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 5890, August 2010

[RFC6365]

Hoffman, P., and Klensin, J., "Terminology Used in Internationalization in the IETF", RFC 6365, September 2011

7.2. Informative References

[RFC2194]

Aboba, B., Lu, J., Alsop, J., Ding, J., and W. Wang, "Review of Roaming Implementations", RFC 2194, September 1997.

[RFC2341]

Valencia, A., Littlewood, M., and T. Kolar, "Cisco Layer Two Forwarding (Protocol) "L2F"", RFC 2341, May 1998.

[RFC2637]

Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol", RFC 2637, July 1999.

[RFC2661]

Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, August 1999.

[RFC2865]

Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC2866]

Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC3579]

Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RFC3748]

Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC4282]

Aboba, B. et al., "The Network Access Identifier", RFC 4282, December 2005.

[RFC4301]

Kent, S. and S. Keo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5335]

Y. Abel, Ed., "Internationalized Email Headers", RFC 5335, September 2008.

[EDUROAM]

<http://eduroam.org>, "eduroam (EDUcational ROAMing)"

[RFC5891]

Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891

[RFC6055]

Thaler, D., et al, "IAB Thoughts on Encodings for Internationalized Domain Names", RFC 6055, February 2011.

[RFC6733]

V. Fajardo, Ed., et al, "Diameter Base Protocol", RFC 6733, October 2012.

[RFC6912]

Sullivan, A., et al, "Principles for Unicode Code Point Inclusion in Labels in the DNS", RFC 6912, April 2013.

Acknowledgments

The initial text for this document was [RFC4282], which was then heavily edited. The original authors of [RFC4282] were Bernard Aboba, Mark A. Beadles, Jari Arkko, and Pasi Eronen.

The ABNF validator at <http://www.apps.ietf.org/abnf.html> was used to verify the syntactic correctness of the ABNF in Section 2.

Appendix A - Changes from RFC4282

This document contains the following updates with respect to the previous NAI definition in RFC 4282 [RFC4282]:

- * The formal syntax in Section 2.1 has been updated to forbid non-UTF8 characters. e.g. characters with the "high bit" set.
- * The formal syntax in Section 2.1 has been updated to allow UTF-8 in the "realm" portion of the NAI.
- * The formal syntax in [RFC4282] Section 2.1 applied to the NAI after it was "internationalized" via the ToAscii function. The contents of the NAI before it was "internationalized" were left indeterminate. This document updates the formal syntax to define an internationalized form of the NAI, and forbids the use of the ToAscii function for NAI "internationalization".
- * The grammar for the user and realm portion is based on a combination of the "nai" defined in [RFC4282] Section 2.1, and the "utf8-addr-spec" defined in [RFC5335] Section 4.4.
- * All use of the ToAscii function has been moved to normal requirements on DNS implementations when realms are used as the basis for DNS lookups. This involves no changes to the existing DNS infrastructure.
- * The discussions on internationalized character sets in Section 2.4 have been updated. The suggestion to use the ToAscii function for realm comparisons has been removed. No AAA system has implemented these suggestions, so this change should have no operational impact.
- * The section "Routing inside of AAA Systems" section is new in this document. The concept of a "local AAA routing table" is also new, although it accurately describes the functionality of wide-spread implementations.
- * The "Compatibility with EMail Usernames" and "Compatibility with DNS" sections have been revised and updated. We now note that the ToAscii function is suggested to be used only when a realm name is used for DNS lookups, and even then the function is only used by a resolving API on the local system, and even then we recommend that only the home network perform this conversion.
- * The "Realm Construction" section has been updated to note that editing of the NAI is NOT RECOMMENDED.

- * The "Examples" section has been updated to remove the instance of the IDN being converted to ASCII. This behavior is now forbidden.

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org

RADIUS EXTensions Working Group
Internet-Draft
Updates: RFC6929 (if approved)
Intended status: Experimental
Expires: January 5, 2015

A. Perez-Mendez
R. Marin-Lopez
F. Pereniguez-Garcia
G. Lopez-Millan
University of Murcia
D. Lopez
Telefonica I+D
A. DeKok
Network RADIUS
July 4, 2014

Support of fragmentation of RADIUS packets
draft-ietf-radext-radius-fragmentation-07

Abstract

The Remote Authentication Dial-In User Service (RADIUS) protocol is limited to a total packet size of 4096 octets. Provisions exist for fragmenting large amounts of authentication data across multiple packets, via Access-Challenge. No similar provisions exist for fragmenting large amounts of authorization data. This document specifies how existing RADIUS mechanisms can be leveraged to provide that functionality. These mechanisms are largely compatible with existing implementations, and are designed to be invisible to proxies, and "fail-safe" to legacy clients and servers.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
2. Scope of this document	4
3. Overview	7
4. Fragmentation of packets	9
4.1. Pre-authorization	10
4.2. Post-authorization	14
5. Chunk size	17
6. Allowed large packet size	18
7. Handling special attributes	19
7.1. Proxy-State attribute	19
7.2. State attribute	20
7.3. Service-Type attribute	21
7.4. Rebuilding the original large packet	21
8. New flag T field for the Long Extended Type attribute definition	22
9. New attribute definition	22
9.1. Frag-Status attribute	22
9.2. Proxy-State-Len attribute	23
9.3. Table of attributes	24
10. Operation with proxies	25
10.1. Legacy proxies	25
10.2. Updated proxies	25
11. Operational considerations	27
11.1. Flag T	27
11.2. Violation of RFC2865	28
11.3. Proxying based on User-Name	28
11.4. Transport behaviour	28
12. Security Considerations	29
13. IANA Considerations	29
14. Acknowledgements	30
15. References	30
15.1. Normative References	30
15.2. Informative References	31
Authors' Addresses	31

1. Introduction

The RADIUS [RFC2865] protocol carries authentication, authorization, and accounting information between a Network Access Server (NAS) and an Authentication Server (AS). Information is exchanged between the NAS and the AS through RADIUS packets. Each RADIUS packet is composed of a header, and zero or more attributes, up to a maximum packet size of 4096 octets. The protocol is a request/response protocol, as described in the operational model ([RFC6158], Section 3.1).

The above packet size limitation mean that peers desiring to send large amounts of data must fragment it across multiple packets. For example, RADIUS-EAP [RFC3579] defines how an EAP exchange occurs across multiple Access-Request / Access-Challenge sequences. No such exchange is possible for accounting or authorization data. [RFC6158] Section 3.1 suggests that exchanging large amounts authorization data is unnecessary in RADIUS. Instead, the data should be referenced by name. This requirement allows large policies to be pre-provisioned, and then referenced in an Access-Accept. In some cases, however, the authorization data sent by the server is large and highly dynamic. In other cases, the NAS needs to send large amounts of authorization data to the server. Both of these cases are un-met by the requirements in [RFC6158]. As noted in that document, the practical limit on RADIUS packet sizes is governed by the Path MTU (PMTU), which may be significantly smaller than 4096 octets. The combination of the two limitations means that there is a pressing need for a method to send large amounts of authorization data between NAS and AS, with no accompanying solution.

[RFC6158] recommends three approaches for the transmission of large amount of data within RADIUS. However, they are not applicable to the problem statement of this document for the following reasons:

- o The first approach does not talk about large amounts of data sent from the NAS to a server. Leveraging EAP (request/challenge) to send the data is not feasible, as EAP already fills packet to PMTU, and not all authentications use EAP. Moreover, as noted for NAS-Filter-Rule ([RFC4849]), this approach does entirely solve the problem of sending large amounts of data from a server to a NAS.
- o The second approach is not usable either, as using names rather than values is difficult when the nature of the data to be sent is highly dynamic (e.g. SAML sentences or NAS-Filter-Rule attributes). URLs could be used as a pointer to the location of the actual data, but their use would require them to be (a) dynamically created and modified, (b) securely accessed and (c) accessible from remote systems. Satisfying these constraints

would require the modification of several networking systems (e.g. firewalls and web servers). Furthermore, the set up of an additional trust infrastructure (e.g. PKI) would be required to allow secure retrieving of the information from the web server.

- o PMTU discovery does not solve the problem, as it does not allow to send data larger than the minimum of (PMTU or 4096) octets.

This document provides a mechanism to allow RADIUS peers to exchange large amounts of authorization data exceeding the 4096 octet limit, by fragmenting it across several client/server exchanges. The proposed solution does not impose any additional requirements to the RADIUS system administrators (e.g. need to modify firewall rules, set up web servers, configure routers, or modify any application server). It maintains compatibility with intra-packet fragmentation mechanisms (like those defined in [RFC3579] or in [RFC6929]). It is also transparent to existing RADIUS proxies, which do not implement this specification. The only systems needing to implement the draft are the ones which either generate, or consume the fragmented data being transmitted. Intermediate proxies just pass the packets without changes. Nevertheless, if a proxy supports this specification, it may re-assemble the data in order to either examine and/or modify it.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. When these words appear in lower case, they have their natural language meaning.

2. Scope of this document

This specification describes how a RADIUS client and a RADIUS server can exchange data exceeding the 4096 octet limit imposed by one packet. However, the mechanism described in this specification MUST NOT be used to exchange more than 100K of data. It has not been designed to substitute for stream-oriented transport protocols, such as TCP or SCTP. Experience shows that attempts to transport bulk data across the Internet with UDP will inevitably fail, unless they re-implement all of the behavior of TCP. The underlying design of RADIUS lacks the proper retransmission policies or congestion control mechanisms which would make it a competitor to TCP.

Therefore, RADIUS/UDP transport is by design unable to transport bulk data. It is both undesired and impossible to change the protocol at this point in time. This specification is intended to allow the

transport of more than 4096 octets of data through existing RADIUS/UDP proxies. Other solutions such as RADIUS/TCP MUST be used when a "green field" deployment requires the transport of bulk data.

Section 6, below, describes with further details the reasoning for this limitation, and recommends administrators to adjust it according to the specific capabilities of their existing systems in terms of memory and processing power.

Moreover, its scope is limited to the exchange of authorization data, as other exchanges do not require of such a mechanism. In particular, authentication exchanges have already been defined to overcome this limitation (e.g. RADIUS-EAP). Moreover, as they represent the most critical part of a RADIUS conversation, it is preferable to not introduce any modification to their operation that may affect existing equipment.

There is no need to fragment accounting packets either. While the accounting process can send large amounts of data, that data is typically composed of many small updates. That is, there is no demonstrated need to send indivisible blocks of more than 4K of data. The need to send large amounts of data per user session often originates from the need for flow-based accounting. In this use-case, the client may send accounting data for many thousands of flows, where all those flows are tied to one user session. The existing Acct-Multi-Session-Id attribute defined in [RFC2866] Section 5.11 has been proven to work here.

Similarly, there is no need to fragment CoA packets. Instead, the CoA client MUST send a CoA-Request packet containing session identification attributes, along with Service-Type = Additional-Authorization, and a State attribute. Implementations not supporting fragmentation will respond with a CoA-NAK, and an Error-Cause of Unsupported-Service.

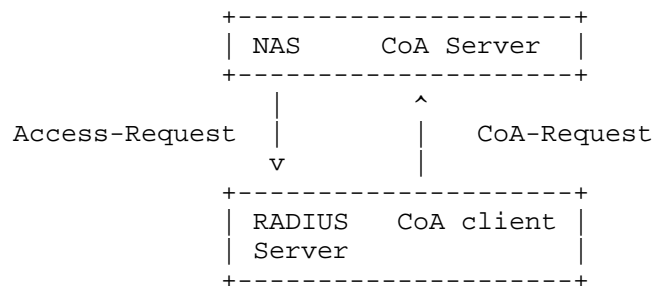
The above requirement does not assume that the CoA client and the RADIUS server are co-located. They may, in fact be run on separate parts of the infrastructure, or even by separate administrators. There is, however, a requirement that the two communicate. We can see that the CoA client needs to send session identification attributes in order to send CoA packets. These attributes cannot be known a priori by the CoA client, and can only come from the RADIUS server. Therefore, even when the two systems are not co-located, they must be able to communicate in order to operate in unison. The alternative is for the two systems to have differing views of the users authorization parameters, which is a security disaster.

This specification does not allow for fragmentation of CoA packets.

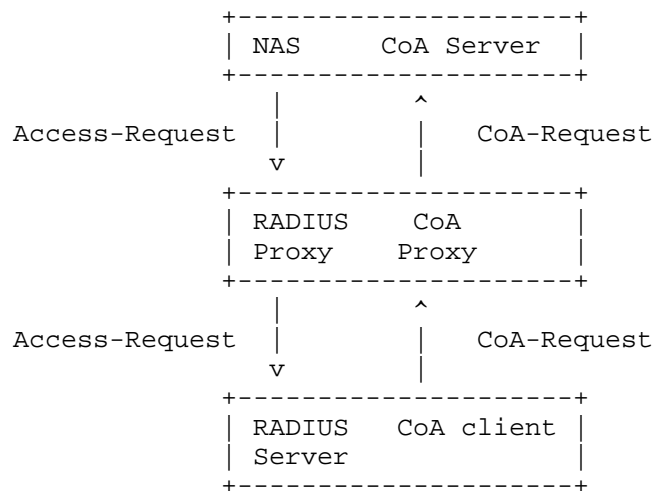
Allowing for fragmented CoA packets would involve changing multiple parts of the RADIUS protocol, with the corresponding possibility for implementation issues, mistakes, etc.

Where CoA clients (i.e. RADIUS servers) need to send large amounts of authorization data to a CoA server (i.e. NAS), they need only send a minimal CoA-Request packet, containing Service-Type of Authorize-Only, as per RFC 5176, along with session identification attributes. This CoA packet serves as a signal to the NAS that the users' session requires re-authorization. When the NAS re-authorizes the user via Access-Request, the RADIUS server can perform fragmentation, and send large amounts of authorization data to the NAS.

The assumption in the above scenario is that the CoA client and RADIUS server are co-located, or at least strongly coupled. That is, the path from CoA client to CoA server SHOULD be the exact reverse of the path from NAS to RADIUS server. The following diagram will hopefully clarify the roles:



Where there is a proxy involved:



That is, the RADIUS and CoA subsystems at each hop are strongly connected. Where they are not strongly connected, it will be impossible to use CoA-Request packets to transport large amounts of authorization data.

This design is more complicated than allowing for fragmented CoA packets. However, the CoA client and the RADIUS server must communicate even when not using this specification. We believe that standardizing that communication, and using one method for exchange of large data is preferred to unspecified communication methods and multiple ways of achieving the same result. If we were to allow fragmentation of data over CoA packets, the size and complexity of this specification would increase significantly.

The above requirement solves a number of issues. It clearly separates session identification from authorization. Without this separation, it is difficult to both identify a session, and change its authorization using the same attribute. It also ensures that the authorization process is the same for initial authentication, and for CoA.

3. Overview

Authorization exchanges can occur either before or after end user authentication has been completed. An authorization exchange before authentication allows a RADIUS client to provide the RADIUS server with information that MAY modify how the authentication process will be performed (e.g. it may affect the selection of the EAP method). An authorization exchange after authentication allows the RADIUS

server to provide the RADIUS client with information about the end user, the results of the authentication process and/or obligations to be enforced. In this specification we refer to the "pre-authorization" as the exchange of authorization information before the end user authentication has started (from the NAS to the AS), whereas the term "post-authorization" is used to refer to an authorization exchange happening after this authentication process (from the AS to the NAS).

In this specification we refer to the "size limit" as the practical limit on RADIUS packet sizes. This limit is the minimum of 4096 octets, and the current PMTU. We define below a method which uses Access-Request and Access-Accept in order to exchange fragmented data. The NAS and server exchange a series of Access-Request / Access-Accept packets, until such time as all of the fragmented data has been transported. Each packet contains a Frag-Status attribute which lets the other party know if fragmentation is desired, ongoing, or finished. Each packet may also contain the fragmented data, or instead be an "ACK" to a previous fragment from the other party. Each Access-Request contains a User-Name attribute, allowing the packet to be proxied if necessary (see Section 10.1). Each Access-Request may also contain a State attribute, which serves to tie it to a previous Access-Accept. Each Access-Accept contains a State attribute, for use by the NAS in a later Access-Request. Each Access-Accept contains a Service-Type attribute with the "Additional-Authorization" value. This indicates that the service being provided is part of a fragmented exchange, and that the Access-Accept should not be interpreted as providing network access to the end user.

When a RADIUS client or server need to send data that exceeds the size limit, the mechanism proposed in this document is used. Instead of encoding one large RADIUS packet, a series of smaller RADIUS packets of the same type are encoded. Each smaller packet is called a "chunk" in this specification, in order to distinguish it from traditional RADIUS packets. The encoding process is a simple linear walk over the attributes to be encoded. This walk preserves the order of the attributes of the same type, as required by [RFC2865]. The number of attributes encoded in a particular chunk depends on the size limit, the size of each attribute, the number of proxies between client and server, and the overhead for fragmentation signalling attributes. Specific details are given in Section 5. A new attribute called Frag-Status (Section 9.1) signals the fragmentation status.

After the first chunk is encoded, it is sent to the other party. The packet is identified as a chunk via the Frag-Status attribute. The other party then requests additional chunks, again using the Frag-Status attribute. This process is repeated until all the attributes

have been sent from one party to the other. When all the chunks have been received, the original list of attributes is reconstructed and processed as if it had been received in one packet.

When multiple chunks are sent, a special situation may occur for Extended Type attributes as defined in [RFC6929]. The fragmentation process may split a fragmented attribute across two or more chunks, which is not permitted by that specification. We address this issue by using the newly defined flag "T" in the Reserved field of the "Long Extended Type" attribute format (see Section 8 for further details on this flag).

This last situation is expected to be the most common occurrence in chunks. Typically, packet fragmentation will occur as a consequence of a desire to send one or more large (and therefore fragmented) attributes. The large attribute will likely be split into two or more pieces. Where chunking does not split a fragmented attribute, no special treatment is necessary.

The setting of the "T" flag is the only case where the chunking process affects the content of an attribute. Even then, the "Value" fields of all attributes remain unchanged. Any per-packet security attributes such as Message-Authenticator are calculated for each chunk independently. There are neither integrity nor security checks performed on the "original" packet.

Each RADIUS packet sent or received as part of the chunking process MUST be a valid packet, subject to all format and security requirements. This requirement ensures that a "transparent" proxy not implementing this specification can receive and send compliant packets. That is, a proxy which simply forwards packets without detailed examination or any modification will be able to proxy "chunks".

4. Fragmentation of packets

When the NAS or the AS desires to send a packet that exceeds the size limit, it is split into chunks and sent via multiple client/server exchanges. The exchange is indicated via the Frag-Status attribute, which has value More-Data-Pending for all but the last chunk of the series. The chunks are tied together via the State attribute.

The delivery of a large fragmented RADIUS packet with authorization data can happen before or after the end user has been authenticated by the AS. We can distinguish two phases:

1. Pre-authorization. In this phase, the NAS can send a large packet with authorization information to the AS before the end user is authenticated.
2. Post-authorization. In this phase, the AS can send a large packet with authorization data to the NAS after the end user has been authenticated.

The following subsections describe how to perform fragmentation for packets for these two phases, pre-authorization and post-authorization. We give the packet type, along with a RADIUS Identifier, to indicate that requests and responses are connected. We then give a list of attributes. We do not give values for most attributes, as we wish to concentrate on the fragmentation behaviour, rather than packet contents. Attribute values are given for attributes relevant to the fragmentation process. Where "long extended" attributes are used, we indicate the M (More) and T (Truncation) flags as optional square brackets after the attribute name. As no "long extended" attributes have yet been defined, we use example attributes, named as "Example-Long-1", etc. The maximum chunk size is established in term of number of attributes (11), for sake of simplicity.

4.1. Pre-authorization

When the client needs to send a large amount of data to the server, the data to be sent is split into chunks and sent to the server via multiple Access-Request / Access-Accept exchanges. The example below shows this exchange.

The following is an Access-Request which the NAS intends to send to a server. However, due to a combination of issues (PMTU, large attributes, etc.), the content does not fit into one Access-Request packet.

```
Access-Request
  User-Name
  NAS-Identifier
  Calling-Station-Id
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1
  Example-Long-2 [M]
  Example-Long-2 [M]
  Example-Long-2
```

Figure 1: Desired Access-Request

The NAS therefore must send the attributes listed above in a series of chunks. The first chunk contains eight (8) attributes from the original Access-Request, and a Frag-Status attribute. Since last attribute is "Example-Long-1" with the "M" flag set, the chunking process also sets the "T" flag in that attribute. The Access-Request is sent with a RADIUS Identifier field having value 23. The Frag-Status attribute has value More-Data-Pending, to indicate that the NAS wishes to send more data in a subsequent Access-Request. The NAS also adds a Service-Type attribute, which indicates that it is part of the chunking process. The packet is signed with the Message-Authenticator attribute, completing the maximum number of attributes (11).

```
Access-Request (ID = 23)
  User-Name
  NAS-Identifier
  Calling-Station-Id
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [MT]
  Frag-Status = More-Data-Pending
  Service-Type = Additional-Authorization
  Message-Authenticator
```

Figure 2: Access-Request (chunk 1)

Compliant servers (i.e. servers implementing fragmentation) receiving

this packet will see the Frag-Status attribute, and postpone all authorization and authentication handling until all of the chunks have been received. This postponement also affects to the verification that the Access-Request packet contains some kind of authentication attribute (e.g. User-Password, CHAP-Password, State or other future attribute), as required by [RFC2865] (see Section 11.2 for more information on this).

Non-compliant servers (i.e. servers not implementing fragmentation) should also see the Service-Type requesting provisioning for an unknown service, and return Access-Reject. Other non-compliant servers may return an Access-Reject, Access-Challenge, or an Access-Accept with a particular Service-Type other than Additional-Authorization. Compliant NAS implementations MUST treat these responses as if they had received Access-Reject instead.

Compliant servers who wish to receive all of the chunks will respond with the following packet. The value of the State here is arbitrary, and serves only as a unique token for example purposes. We only note that it MUST be temporally unique to the server.

```
Access-Accept (ID = 23)
  Frag-Status = More-Data-Request
  Service-Type = Additional-Authorization
  State = 0xabc00001
  Message-Authenticator
```

Figure 3: Access-Accept (chunk 1)

The NAS will see this response, and use the RADIUS Identifier field to associate it with an ongoing chunking session. Compliant NASes will then continue the chunking process. Non-compliant NASes will never see a response such as this, as they will never send a Frag-Status attribute. The Service-Type attribute is included in the Access-Accept in order to signal that the response is part of the chunking process. This packet therefore does not provision any network service for the end user.

The NAS continues the process by sending the next chunk, which includes an additional six (6) attributes from the original packet. It again includes the User-Name attribute, so that non-compliant proxies can process the packet (see Section 10.1). It sets the Frag-Status attribute to More-Data-Pending, as more data is pending. It includes a Service-Type for reasons described above. It includes the State attribute from the previous Access-accept. It signs the packet with Message-Authenticator, as there are no authentication attributes in the packet. It uses a new RADIUS Identifier field.


```
Access-Request (ID = 181)
  User-Name
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1
  Example-Long-2 [M]
  Example-Long-2 [MT]
  Frag-Status = More-Data-Pending
  Service-Type = Additional-Authorization
  State = 0xabc000001
  Message-Authenticator
```

Figure 4: Access-Request (chunk 2)

Compliant servers receiving this packet will see the Frag-Status attribute, and look for a State attribute. Since one exists and it matches a State sent in an Access-Accept, this packet is part of a chunking process. The server will associate the attributes with the previous chunk. Since the Frag-Status attribute has value More-Data-Request, the server will respond with an Access-Accept as before. It MUST include a State attribute, with a value different from the previous Access-Accept. This State MUST again be globally and temporally unique.

```
Access-Accept (ID = 181)
  Frag-Status = More-Data-Request
  Service-Type = Additional-Authorization
  State = 0xdef00002
  Message-Authenticator
```

Figure 5: Access-Accept (chunk 2)

The NAS will see this response, and use the RADIUS Identifier field to associate it with an ongoing chunking session. The NAS continues the chunking process by sending the next chunk, with the final attribute(s) from the original packet, and again includes the original User-Name attribute. The Frag-Status attribute is not included in the next Access-Request, as no more chunks are available for sending. The NAS includes the State attribute from the previous Access-accept. It signs the packet with Message-Authenticator, as there are no authentication attributes in the packet. It again uses a new RADIUS Identifier field.

```
Access-Request (ID = 241)
  User-Name
  Example-Long-2
  State = 0xdef00002
  Message-Authenticator
```

Figure 6: Access-Request (chunk 3)

On reception of this last chunk, the server matches it with an ongoing session via the State attribute, and sees that there is no Frag-Status attribute present. It then processes the received attributes as if they had been sent in one RADIUS packet. See Section 7.4 for further details of this process. It generates the appropriate response, which can be either Access-Accept or Access-Reject. In this example, we show an Access-Accept. The server MUST send a State attribute, which permits link the received data with the authentication process.

```
Access-Accept (ID = 241)
  State = 0x98700003
  Message-Authenticator
```

Figure 7: Access-Accept (chunk 3)

The above example shows in practice how the chunking process works. We re-iterate the implementation and security requirements here.

Each chunk is a valid RADIUS packet (see Section 11.2 for some considerations about this), and all RADIUS format and security requirements MUST be followed before any chunking process is applied.

Every chunk except for the last one from a NAS MUST include a Frag-Status attribute, with value More-Data-Pending. The last chunk MUST NOT contain a Frag-Status attribute. Each chunk except for the last from a NAS MUST include a Service-Type attribute, with value Additional-Authorization. Each chunk MUST include a User-Name attribute, which MUST be identical in all chunks. Each chunk except for the first one from a NAS MUST include a State attribute, which MUST be copied from a previous Access-Accept.

Each Access-Accept MUST include a State attribute. The value for this attribute MUST change in every new Access-Accept, and MUST be globally and temporally unique.

4.2. Post-authorization

When the AS wants to send a large amount of authorization data to the NAS after authentication, the operation is very similar to the pre-

authorization one. The presence of Service-Type = Additional-Authorization attribute ensures that a NAS not supporting this specification will treat that unrecognized Service-Type as though an Access-Reject had been received instead ([RFC2865] Section 5.6). If the original large Access-Accept packet contained a Service-Type attribute, it will be included with its original value in the last transmitted chunk, to avoid confusion with the one used for fragmentation signalling. It is strongly RECOMMENDED that servers include a State attribute on their original Access-Accept packets, even if fragmentation is not taking place, to allow the client to send additional authorization data in subsequent exchanges. This State attribute would be included in the last transmitted chunk, to avoid confusion with the ones used for fragmentation signalling.

Client supporting this specification MUST include a Frag-Status = Fragmentation-Supported attribute in the first Access-Request sent to the server, in order to indicate they would accept fragmented data from the sever. This is not required if pre-authorization process was carried out, as it is implicit.

The following is an Access-Accept which the AS intends to send to a client. However, due to a combination of issues (PMTU, large attributes, etc.), the content does not fit into one Access-Accept packet.

```
Access-Accept
  User-Name
  EAP-Message
  Service-Type(Login)
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1
  Example-Long-2 [M]
  Example-Long-2 [M]
  Example-Long-2
  State = 0xcba00003
```

Figure 8: Desired Access-Accept

The AS therefore must send the attributes listed above in a series of chunks. The first chunk contains seven (7) attributes from the original Access-Accept, and a Frag-Status attribute. Since last

attribute is "Example-Long-1" with the "M" flag set, the chunking process also sets the "T" flag in that attribute. The Access-Accept is sent with a RADIUS Identifier field having value 30 corresponding to a previous Access-Request not depicted. The Frag-Status attribute has value More-Data-Pending, to indicate that the AS wishes to send more data in a subsequent Access-Accept. The AS also adds a Service-Type attribute with value Additional-Authorization, which indicates that it is part of the chunking process. Note that the original Service-Type is not included in this chunk. Finally, a State attribute is included to allow matching subsequent requests with this conversation, and the packet is signed with the Message-Authenticator attribute, completing the maximum number of attributes of 11.

```
Access-Accept (ID = 30)
  User-Name
  EAP-Message
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [MT]
  Frag-Status = More-Data-Pending
  Service-Type = Additional-Authorization
  State = 0xcba00004
  Message-Authenticator
```

Figure 9: Access-Accept (chunk 1)

Compliant clients receiving this packet will see the Frag-Status attribute, and suspend all authorization and authentication handling until all of the chunks have been received. Non-compliant clients should also see the Service-Type indicating the provisioning for an unknown service, and will treat it as an Access-Reject.

Clients who wish to receive all of the chunks will respond with the following packet, where the value of the State attribute is taken from the received Access-Accept. They also include the User-Name attribute so that non-compliant proxies can process the packet (Section 10.1).

```
Access-Request (ID = 131)
  User-Name
  Frag-Status = More-Data-Request
  Service-Type = Additional-Authorization
  State = 0xcba00004
  Message-Authenticator
```

Figure 10: Access-Request (chunk 1)

The AS receives this request, and uses the State attribute to associate it with an ongoing chunking session. Compliant ASes will then continue the chunking process. Non-compliant ASes will never see a response such as this, as they will never send a Frag-Status attribute.

The AS continues the chunking process by sending the next chunk, with the final attribute(s) from the original packet. The value of the Identifier field is taken from the received Access-Request. A Frag-Status attribute is not included in the next Access-Accept, as no more chunks are available for sending. The AS includes the original State attribute to allow the client to send additional authorization data. The original Service-Type attribute is included as well.

```
Access-Accept (ID = 131)
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1 [M]
  Example-Long-1
  Example-Long-2 [M]
  Example-Long-2 [M]
  Example-Long-2
  Service-Type = Login
  State = 0xfda000003
  Message-Authenticator
```

Figure 11: Access-Accept (chunk 2)

On reception of this last chunk, the client matches it with an ongoing session via the Identifier field, and sees that there is no Frag-Status attribute present. It then processes the received attributes as if they had been sent in one RADIUS packet. See Section 7.4 for further details of this process.

5. Chunk size

In an ideal scenario, each intermediate chunk would be exactly the size limit in length. In this way, the number of round trips required to send a large packet would be optimal. However, this is not possible for several reasons.

1. RADIUS attributes have a variable length, and must be included completely in a chunk. Thus, it is possible that, even if there is some free space in the chunk, it is not enough to include the next attribute. This can generate up to 254 octets of spare space on every chunk.

2. RADIUS fragmentation requires the introduction of some extra attributes for signalling. Specifically, a Frag-Status attribute (7 octets) is included on every chunk of a packet, except the last one. A RADIUS State attribute (from 3 to 255 octets) is also included in most chunks, to allow the server to bind an Access-Request with a previous Access-Challenge. User-Name attributes (from 3 to 255 octets) are introduced on every chunk the client sends as they are required by the proxies to route the packet to its destination. Together, these attributes can generate from up to 13 to 517 octets of signalling data, reducing the amount of payload information that can be sent on each chunk.
 3. RADIUS packets SHOULD be adjusted to avoid exceeding the network MTU. Otherwise, IP fragmentation may occur, having undesirable consequences. Hence, maximum chunk size would be decreased from 4096 to the actual MTU of the network.
 4. The inclusion of Proxy-State attributes by intermediary proxies can decrease the availability of usable space into the chunk. This is described with further detail in Section 7.1.
6. Allowed large packet size

There are no provisions for signalling how much data is to be sent via the fragmentation process as a whole. It is difficult to define what is meant by the "length" of any fragmented data. That data can be multiple attributes, which includes RADIUS attribute header fields. Or it can be one or more "large" attributes (more than 256 octets in length). Proxies can also filter these attributes, to modify, add, or delete them and their contents. These proxies act on a "packet by packet" basis, and cannot know what kind of filtering actions they take on future packets. As a result, it is impossible to signal any meaningful value for the total amount of additional data.

Unauthenticated clients are permitted to trigger the exchange of large amounts of fragmented data between the NAS and the AS, having the potential to allow Denial of Service (DoS) attacks. An attacker could initiate a large number of connections, each of which requests the server to store a large amount of data. This data could cause memory exhaustion on the server, and result in authentic users being denied access. It is worth noting that authentication mechanisms are already designed to avoid exceeding the size limit.

Hence, implementations of this specification MUST limit the total amount of data they send and/or receive via this specification. Its default value SHOULD be 100K. Any more than this may turn RADIUS into

a generic transport protocol, which is undesired. This limit SHOULD be configurable, so that it can be changed if necessary.

Implementations of this specification MUST limit the total number of round trips used during the fragmentation process. Its default value SHOULD be to 25. Any more than this may indicate an implementation error, misconfiguration, or a denial of service (DoS) attack. This limit SHOULD be configurable, so that it can be changed if necessary.

For instance, let's imagine the RADIUS server wants to transport an SAML assertion which is 15000 octets long, to the RADIUS client. In this hypothetical scenario, we assume there are 3 intermediate proxies, each one inserting a Proxy-State attribute of 20 octets. Also we assume the State attributes generated by the RADIUS server have a size of 6 octets, and the User-Name attribute take 50 octets. Therefore, the amount of free space in a chunk for the transport of the SAML assertion attributes is: Total (4096) - RADIUS header (20) - User-Name (50 octets) - Frag-Status (7 octets) - Service-Type (6 octets) - State (6 octets) - Proxy-State (20 octets) - Proxy-State (20) - Proxy-State (20) - Message-Authenticator (18 octets), resulting in a total of 3929 octets, that is, 15 attributes of 255 bytes.

According to [RFC6929], a Long-Extended-Type provides a payload of 251 octets. Therefore, the SAML assertion described above would result into 60 attributes, requiring of 4 round-trips to be completely transmitted.

7. Handling special attributes

7.1. Proxy-State attribute

RADIUS proxies may introduce Proxy-State attributes into any Access-Request packet they forward. Should they be unable to add this information to the packet, they may silently discard forwarding it to its destination, leading to DoS situations. Moreover, any Proxy-State attribute received by a RADIUS server in an Access-Request packet MUST be copied into the reply packet to it. For these reasons, Proxy-State attributes require a special treatment within the packet fragmentation mechanism.

When the RADIUS server replies to an Access-Request packet as part of a conversation involving a fragmentation (either a chunk or a request for chunks), it MUST include every Proxy-State attribute received into the reply packet. This means that the server MUST take into account the size of these Proxy-State attributes in order to calculate the size of the next chunk to be sent.

However, while a RADIUS server will always know how much space MUST be left on each reply packet for Proxy-State attributes (as they are directly included by the RADIUS server), a RADIUS client cannot know this information, as Proxy-State attributes are removed from the reply packet by their respective proxies before forwarding them back. Hence, clients need a mechanism to discover the amount of space required by proxies to introduce their Proxy-State attributes. In the following we describe a new mechanism to perform such a discovery:

1. When a RADIUS client does not know how much space will be required by intermediate proxies for including their Proxy-State attributes, it SHOULD start using a conservative value (e.g. 1024 octets) as the chunk size.
2. When the RADIUS server receives a chunk from the client, it can calculate the total size of the Proxy-State attributes that have been introduced by intermediary proxies along the path. This information MUST be returned to the client in the next reply packet, encoded into a new attribute called Proxy-State-Len. The server MAY artificially increase this quantity in order to handle with situations where proxies behave inconsistently (e.g. they generate Proxy-State attributes with a different size for each packet), or for situations where intermediary proxies remove Proxy-State attributes generated by other proxies. Increasing this value would make the client to leave some free space for these situations.
3. The RADIUS client SHOULD react upon the reception of this attribute by adjusting the maximum size for the next chunk accordingly. However, as the Proxy-State-Len offers just an estimation of the space required by the proxies, the client MAY select a smaller amount in environments known to be problematic.

7.2. State attribute

This RADIUS fragmentation mechanism makes use of the State attribute to link all the chunks belonging to the same fragmented packet. However, some considerations are required when the RADIUS server is fragmenting a packet that already contains a State attribute for other purposes not related with the fragmentation. If the procedure described in Section 4 is followed, two different State attributes could be included into a single chunk, incurring into two problems. First, [RFC2865] explicitly forbids that more than one State attribute appears into a single packet.

A straightforward solution consists on making the RADIUS server to send the original State attribute into the last chunk of the sequence

(attributes can be re-ordered as specified in [RFC2865]). As the last chunk (when generated by the RADIUS server) does not contain any State attribute due to the fragmentation mechanism, both situations described above are avoided.

Something similar happens when the RADIUS client has to send a fragmented packet that contains a State attribute on it. The client MUST assure that this original State is included into the first chunk sent to the server (as this one never contains any State attribute due to fragmentation).

7.3. Service-Type attribute

This RADIUS fragmentation mechanism makes use of the Service-Type attribute to indicate an Access-Accept packet is not granting access to the service yet, since additional authorization exchange needs to be performed. Similarly to the State attribute, the RADIUS server has to send the original Service-Type attribute into the last Access-Accept of the RADIUS conversation to avoid ambiguity.

7.4. Rebuilding the original large packet

The RADIUS client stores the RADIUS attributes received on each chunk in order to be able to rebuild the original large packet after receiving the last chunk. However, some of these received attributes MUST NOT be stored in this list, as they have been introduced as part of the fragmentation signalling and hence, they are not part of the original packet.

- o State (except the one in the last chunk, if present)
- o Service-Type = Additional-Authorization
- o Frag-Status
- o Proxy-State-Len

Similarly, the RADIUS server MUST NOT store the following attributes as part of the original large packet:

- o State (except the one in the first chunk, if present)
- o Service-Type = Additional-Authorization
- o Frag-Status
- o Proxy-State (except the ones in the last chunk)

- o User-Name (except the one in the first chunk)

8. New flag T field for the Long Extended Type attribute definition

This document defines a new field in the "Long Extended Type" attribute format. This field is one bit in size, and is called "T" for Truncation. It indicates that the attribute is intentionally truncated in this chunk, and is to be continued in the next chunk of the sequence. The combination of the flags "M" and "T" indicates that the attribute is fragmented (flag M), but that all the fragments are not available in this chunk (flag T). Proxies implementing [RFC6929] will see these attributes as invalid (they will not be able to reconstruct them), but they will still forward them as [RFC6929] section 5.2 indicates they SHOULD forward unknown attributes anyway.

As a consequence of this addition, the Reserved field is now 6 bits long (see Section 11.1 for some considerations). The following figure represents the new attribute format.

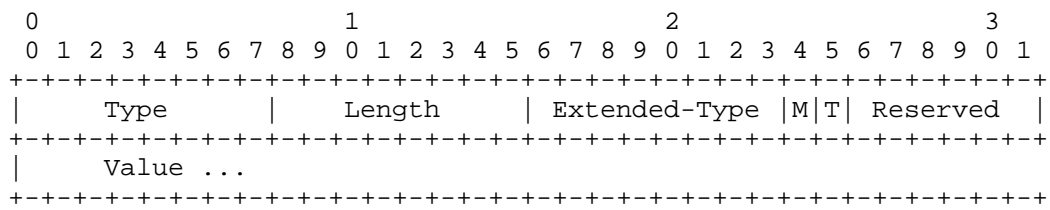


Figure 12: Updated Long Extended Type attribute format

9. New attribute definition

This document proposes the definition of two new extended type attributes, called Frag-Status and Proxy-State-Len. The format of these attributes follows the indications for an Extended Type attribute defined in [RFC6929].

9.1. Frag-Status attribute

This attribute is used for fragmentation signalling, and its meaning depends on the code value transported within it. The following figure represents the format of the Frag-Status attribute.

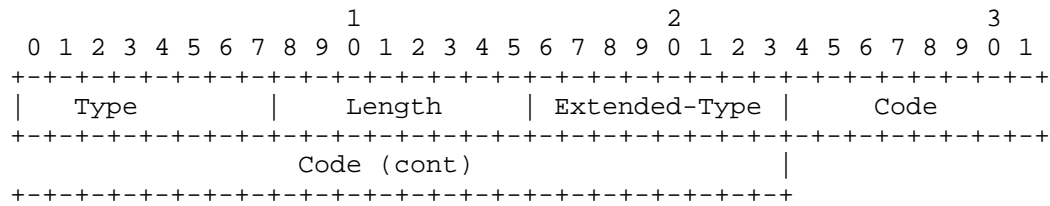


Figure 13: Frag-Status format

Type

To be assigned (TBA)

Length

7

Extended-Type

To be assigned (TBA).

Code

4 byte. Integer indicating the code. The values defined in this specifications are:

- 0 - Reserved
- 1 - Fragmentation-Supported
- 2 - More-Data-Pending
- 3 - More-Data-Request

This attribute MAY be present in Access-Request, Access-Challenge and Access-Accept packets. It MUST NOT be included in Access-Reject packets. Clients supporting this specification MUST include a Frag-Status = Fragmentation-Supported attribute in the first Access-Request sent to the server, in order to indicate they would accept fragmented data from the sever.

9.2. Proxy-State-Len attribute

This attribute indicates to the RADIUS client the length of the Proxy-State attributes received by the RADIUS server. This information is useful to adjust the length of the chunks sent by the RADIUS client. The format of this Proxy-State-Len attribute is the

following:

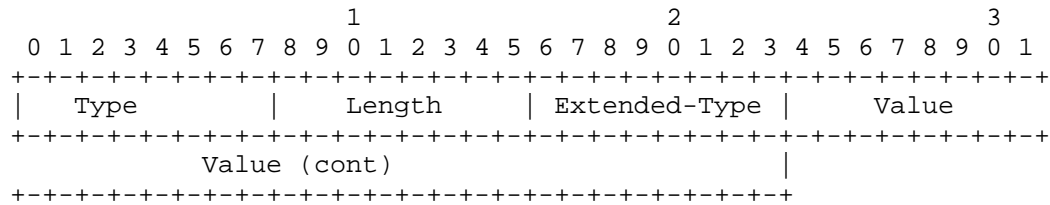


Figure 14: Proxy-State-Len format

Type

To be assigned (TBA)

Length

7

Extended-Type

To be assigned (TBA).

Value

4 octets. Total length (in octets) of received Proxy-State attributes (including headers).

This attribute MAY be present in Access-Challenge and Access-Accept packets. It MUST NOT be included in Access-Request or Access-Reject packets.

9.3. Table of attributes

The following table shows the different attributes defined in this document related with the kind of RADIUS packets where they can be present.

Attribute Name	Kind of packet			
	Req	Acc	Rej	Cha
Frag-Status	0-1	0-1	0	0-1
Proxy-State-Len	0	0-1	0	0-1

10. Operation with proxies

The fragmentation mechanism defined above is designed to be transparent to legacy proxies, as long as they do not want to modify any fragmented attribute. Nevertheless, updated proxies supporting this specification can even modify fragmented attributes.

10.1. Legacy proxies

As every chunk is indeed a RADIUS packet, legacy proxies treat them as the rest of packets, routing them to their destination. Proxies can introduce Proxy-State attributes to Access-Request packets, even if they are indeed chunks. This will not affect how fragmentation is managed. The server will include all the received Proxy-State attributes into the generated response, as described in [RFC2865]. Hence, proxies do not distinguish between a regular RADIUS packet and a chunk.

10.2. Updated proxies

Updated proxies can interact with clients and servers in order to obtain the complete large packet before starting forwarding it. In this way, proxies can manipulate (modify and/or remove) any attribute of the packet, or introduce new attributes, without worrying about crossing the boundaries of the chunk size. Once the manipulated packet is ready, it is sent to the original destination using the fragmentation mechanism (if required). The following example shows how an updated proxy interacts with the NAS to obtain a large Access-Request packet, modify an attribute resulting into a even more large packet, and interacts with the AS to complete the transmission of the modified packet.

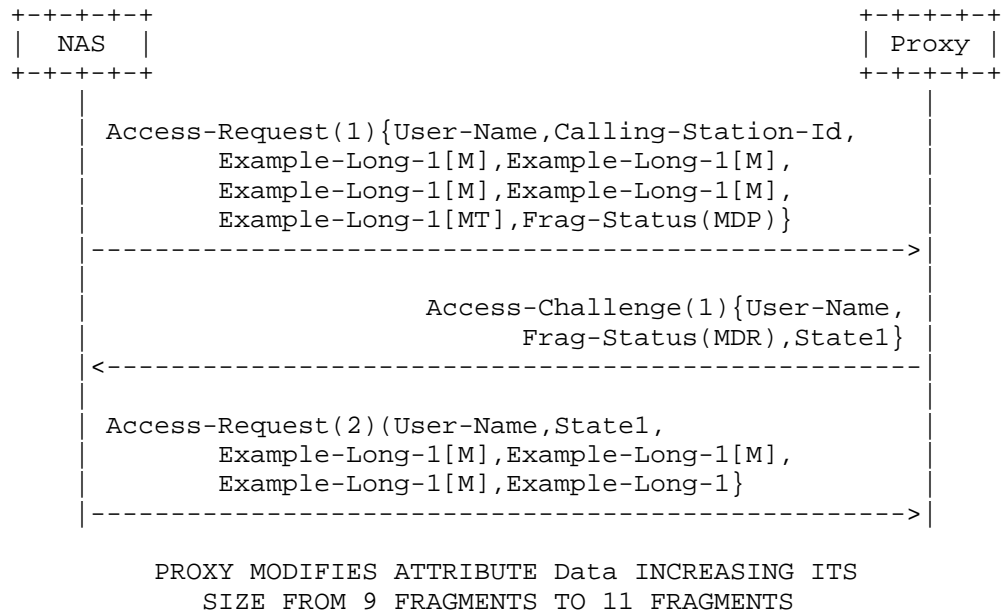


Figure 15: Updated proxy interacts with NAS

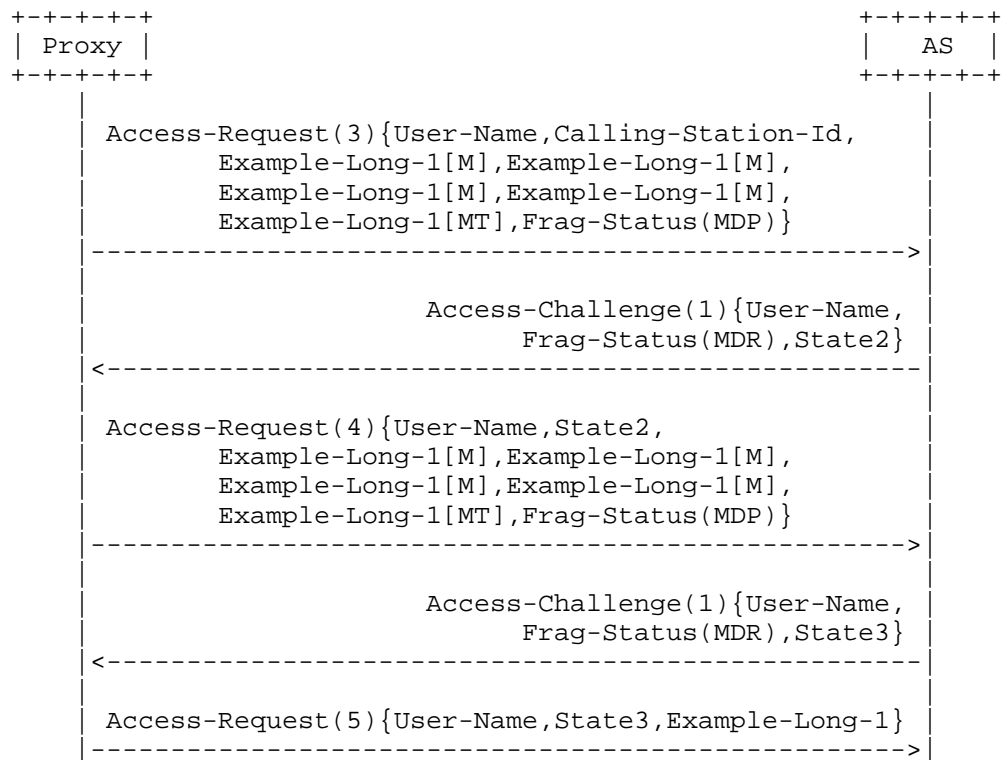


Figure 16: Updated proxy interacts with AS

11. Operational considerations

11.1. Flag T

As described in Section 8, this document modifies the definition of the "Reserved" field of the "Long Extended Type" attribute [RFC6929], by allocating an additional flag "T". The meaning and position of this flag is defined in this document, and nowhere else. This might generate an issue if subsequent specifications want to allocate a new flag as well, as there would be no direct way for them to know which parts of the "Reserved" field have already been defined.

An immediate and reasonable solution for this issue would be declaring that this draft updates [RFC6929]. In this way, [RFC6929] would include an "Updated by" clause that will point readers to this document. However, since this draft belongs to the Experimental track and [RFC6929] belongs to the Standards track, we do not know if including that "Updates" clause would be acceptable.

Another alternative would be creating an IANA registry for the "Reserved" field. However, the working group thinks that would be overkill, as not such a great number of specifications extending that field are expected.

Hence, we have decided to include the "Updates" clause in the document so far.

11.2. Violation of RFC2865

Section 4.1 indicates that all authorization and authentication handling will be postponed until all the chunks have been received. This postponement also affects to the verification that the Access-Request packet contains some kind of authentication attribute (e.g. User-Password, CHAP-Password, State or other future attribute), as required by [RFC2865]. This checking will therefore be delayed until the original large packet has been rebuilt, as some of the chunks may not contain any of them.

The authors acknowledge that this specification violates the "MUST" requirement of [RFC2865] Section 4.1. We note that a proxy which enforces that requirement would be unable to support future RADIUS authentication extensions. Extensions to the protocol would therefore be impossible to deploy. All known implementations have chosen the philosophy of "be liberal in what you accept". That is, they accept traffic which violates the requirement of [RFC2865] Section 4.1. We therefore expect to see no operational issues with this specification. After we gain more operational experience with this specification, it can be re-issued as a standards track document, and update [RFC2865].

11.3. Proxying based on User-Name

This proposal assumes legacy proxies to base their routing decisions on the value of the User-Name attribute. For this reason, every packet sent from the client to the server (either chunks or requests for more chunks) MUST contain a User-Name attribute.

11.4. Transport behaviour

This proposal does not modify the way RADIUS interacts with the underlying transport (UDP). That is, RADIUS keeps following a lock-step behaviour, that requires receiving an explicit acknowledge for each chunk sent. Hence, bursts of traffic which could congest links between peers are not an issue.

12. Security Considerations

As noted in many earlier specifications ([RFC5080], [RFC6158], etc.) RADIUS security is problematic. This specification changes nothing related to the security of the RADIUS protocol. It requires that all Access-Request packets associated with fragmentation are authenticated using the existing Message-Authenticator attribute. This signature prevents forging and replay, to the limits of the existing security.

The ability to send bulk data from one party to another creates new security considerations. Clients and servers may have to store large amounts of data per session. The amount of this data can be significant, leading to the potential for resource exhaustion. We therefore suggest that implementations limit the amount of bulk data stored per session. The exact method for this limitation is implementation-specific. Section 6 gives some indications on what could be reasonable limits.

The bulk data can often be pushed off to storage methods other than the memory of the RADIUS implementation. For example, it can be stored in an external database, or in files. This approach mitigates the resource exhaustion issue, as servers today already store large amounts of accounting data.

13. IANA Considerations

The authors request that Attribute Types and Attribute Values defined in this document be registered by the Internet Assigned Numbers Authority (IANA) from the RADIUS namespaces as described in the "IANA Considerations" section of [RFC3575], in accordance with BCP 26 [RFC5226]. For RADIUS packets, attributes and registries created by this document IANA is requested to place them at <http://www.iana.org/assignments/radius-types>.

In particular, this document defines two new RADIUS attributes, entitled "Frag-Status" and "Proxy-State-Len" (see section 9), assigned values of TBD1 and TBD2 from the Long Extended Space of [RFC2865]:

Tag	Name	Length	Meaning
----	----	-----	-----
TBD1	Frag-Status	7	Signals fragmentation
TBD2	Proxy-State-Len	7	Indicates the length of the received Proxy-State attributes

The Frag-Status attribute also defines a 8-bit "Code" field, for

which the IANA is to create and maintain a new sub-registry entitled "Code values" under the RADIUS "Frag-Status" attribute. Initial values for the RADIUS Frag-Status "Code" registry are given below; future assignments are to be made through "RFC required" [IANA-CONSIDERATIONS]. Assignments consist of a Frag-Status "Code" name and its associated value.

Value	Frag-Status Code Name	Definition
----	-----	-----
0	Reserved	See Section 9.1
1	Fragmentation-Supported	See Section 9.1
2	More-Data-Pending	See Section 9.1
3	More-Data-Request	See Section 9.1
4-255	Unassigned	

Additionally, allocation of a new Service-Type value for "Additional-Authorization" is requested.

Value	Service Type Value	Definition
----	-----	-----
TBA	Additional-Authorization	See section 4.1

14. Acknowledgements

The authors would like to thank the members of the RADEXT working group who have contributed to the development of this specification, either by participating on the discussions on the mailing lists or by sending comments about our draft.

The authors also thank David Cuenca (University of Murcia) for implementing a proof of concept implementation of this draft that has been useful to improve the quality of the specification.

This work has been partly funded by the GEANT GN3+ SA5 and CLASSe (<http://sec.cs.kent.ac.uk/CLASSe/>) projects.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

- [RFC3575] Aboba, B., "IANA Considerations for RADIUS (Remote Authentication Dial In User Service)", RFC 3575, July 2003.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6158] DeKok, A. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011.
- [RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, April 2013.

15.2. Informative References

- [RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
- [RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.
- [RFC4849] Congdon, P., Sanchez, M., and B. Aboba, "RADIUS Filter Rule Attribute", RFC 4849, April 2007.
- [RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, December 2007.

Authors' Addresses

Alejandro Perez-Mendez (Ed.)
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 46 44
Email: alex@um.es

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 85 01
Email: rafa@um.es

Fernando Pereniguez-Garcia
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 78 82
Email: pereniguez@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia, 30100
Spain

Phone: +34 868 88 85 04
Email: gabilm@um.es

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 84
Madrid, 28006
Spain

Phone: +34 913 129 041
Email: diego@tid.es

Alan DeKok
Network RADIUS
15 av du Granier
Meylan, 38240
France

Phone: +34 913 129 041
Email: aland@networkradius.com
URI: <http://networkradius.com>

RADIUS Extensions Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: April 30, 2015

S. Winter
RESTENA
October 27, 2014

Considerations regarding the correct use of EAP-Response/Identity
draft-winter-radext-populating-eapidentity-01

Abstract

There are some subtle considerations for an EAP peer regarding the content of the EAP-Response/Identity packet when authenticating with EAP to an EAP server. This document describes two such considerations and suggests workarounds to the associated problems.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Problem Statement	2
1.2. Requirements Language	2
2. EAP-Response/Identity: Effects on EAP type negotiation . . .	3
3. Character (re-)encoding may be required	4
4. Recommendations for EAP peer implementations	5
5. Privacy Considerations	5
6. Security Considerations	6
7. IANA Considerations	6
8. References	6
8.1. Normative References	6
8.2. Informative References	6

1. Introduction

1.1. Problem Statement

An Extensible Authentication Protocol (EAP, [RFC3748]) conversation between an EAP peer and an EAP server starts with an (optional) request for identity information by the EAP server (EAP-Request/Identity) followed by the peer's response with identity information (EAP-Response/Identity). Only after this identity exchange are EAP types negotiated.

EAP-Response/Identity is sent before EAP type negotiation takes place, but it is not independent of the later-negotiated EAP type. Two entanglements between EAP-Response/Identity and EAP methods' notions of a user identifier are described in this document.

1. The choice of identity to send in EAP-Response/Identity may have detrimental effects on the subsequent EAP type negotiation.
2. Using identity information from the preferred EAP type without thoughtful conversion of character encoding may have detrimental effects on the outcome of the authentication.

The following two chapters describe each of these issues in detail. The last chapter contains recommendations for implementers of EAP peers to avoid these issues.

1.2. Requirements Language

In this document, several words are used to signify the requirements of the specification. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY",

and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. [RFC2119]

2. EAP-Response/Identity: Effects on EAP type negotiation

Assuming the EAP peer's EAP type selection is not the trivial case (i.e. it has more than one configured EAP type for a given network or application, and needs to make a decision which one to use), an issue arises when the configured EAP types are not all configured with the same user identifier.

Issue: if the user identifiers in the set of configured EAP types differ (e.g. have a different [RFC4282] "realm" portion), and the authenticator does not send identity selection hints as per [RFC4284], then EAP type negotiation may be limited to those EAP types which are terminated in the same EAP server. The reason for that is because the information in the EAP-Response/Identity is used for request routing decisions and thus determines the EAP server - a given user identifier may be routed to a server which exclusively serves the matching EAP type. Negotiating another EAP type from the set of configured EAP types during the running EAP conversation is then not possible.

Example:

Assume an EAP peer is configured to support two EAP types:

- o EAP-AKA' [RFC5448] with user identifier imsi@mnc123.mcc123.3gpp-network.org
- o EAP-TTLS [RFC5281] with user identifier john@realm.example

The user connects to hotspot of a roaming consortium which could authenticate him with EAP-TTLS and his john@realm.example identity. The hotspot operator has no business relationship at all with the 3GPP consortium; incoming authentication requests for realms ending in 3gppnetwork.org will be immediately rejected. Identity selection hints are not sent.

Consequence: If the EAP peer consistently chooses the imsi@mnc123.mcc123.3gpp-network.org user identifier as choice for its initial EAP-Response/Identity, the user will be consistently and perpetually rejected, even though in possession of a valid credential for the hotspot.

An EAP peer should always try all options to authenticate. As the example above shows, it may not be sufficient to rely on EAP method negotiation alone to iterate through all configured EAP types and

come to a conclusive outcome of the authentication attempt. Multiple new EAP authentications, each using a different user identifier from the set of configured user identities, may be required to fully iterate through the list of usable identities.

3. Character (re-)encoding may be required

The user identifier as configured in the EAP method configuration is not always suited as user identifier to choose as EAP-Response/Identity. This is trivially true when using tunneled EAP types and configuring anonymous outer identity for the tunneling EAP type. There is at least one additional, non-trivial, case to consider however:

EAP methods define the encoding of their user identifiers; in particular, the encoding of the user identifiers as defined the EAP method may or may not be UTF-8; some EAP methods are even known not to put any encoding restrictions on their user identifiers at all.

It is not the intention of EAP, as a mere method-agnostic container which simply carries EAP types, to restrict an EAP method's choice of encoding of a user identifier. However, there are restrictions in what should be contained in the EAP-Response/Identity: EAP is very often carried over a AAA protocol (e.g over RADIUS as per [RFC3579]). The typical use for the contents of EAP-Response/Identity inside AAA protocols like RADIUS [RFC2865] and Diameter [RFC6733] is to copy the content of EAP-Response/Identity into a "User-Name" attribute; the encoding of the User-Name attribute is required to be UTF-8. EAP-Response/Identity does not carry encoding information itself, so a conversion between a non-UTF-8 encoding and UTF-8 is not possible for the AAA entity doing the EAP-Response/Identity to User-Name copying.

Consequence: If an EAP method's user identifier is not encoded in UTF-8, and the EAP peer verbatimly uses that method's notion of a user identifier for its EAP-Response/Identity field, then the AAA entity is forced to violate its own specification because it has to, but can not use UTF-8 for its own User-Name attribute. If the EAP method configuration sets an outer identity in a non UTF-8 character set, and the EAP peer verbatimly uses that outer identity for its EAP-Response/Identity field, then the same violation occurs.

This jeopardizes the subsequent EAP authentication as a whole; request routing may fail, lead to a wrong destination or introduce routing loops due to differing interpretations of the User-Name in EAP pass-through authenticators and AAA proxies.

4. Recommendations for EAP peer implementations

Where user identifiers between configured EAP types in an EAP peer differ, the EAP peer can not rely on the EAP type negotiation mechanism alone to provide useful results. If an EAP authentication gets rejected, the EAP peer SHOULD re-try the authentication using a different EAP-Response/Identity than before. The EAP peer SHOULD try all user identifiers from the entire set of configured EAP types before declaring final authentication failure.

EAP peers need to maintain state on the encoding of the user identifiers which are used in their locally configured EAP types. When constructing an EAP-Response/Identity from that user identifier, they MUST (re-)encode that user identifier as UTF-8 and use the resulting value for the EAP-Response/Identity. If the EAP type is configured for the use of anonymous outer identities, the desired outer identity MUST also be (re-)encoded in UTF-8 encoding before being put into the EAP-Response/Identity.

5. Privacy Considerations

Because the EAP-Response/Identity content is not encrypted, the backtracking to a new EAP-Response/Identity will systematically reveal all configured identities to intermediate passive listeners on the path between the EAP peer and the EAP server (until one authentication round succeeds).

This additional leakage of identity information is not very significant though because where privacy is considered important, the additional option for identity privacy which is present in most modern EAP methods can be used.

If the EAP peer implementation is certain that all EAP types will be terminated at the same EAP server (e.g. with a corresponding configuration option) then the iteration over all identities can be avoided, because the EAP type negotiation is then sufficient.

If a choice of which identity information to disclose needs to be made by the EAP peer, when iterating through the list of identities the EAP peer SHOULD

- in first priority honour a manually configured order of preference of EAP types, if any

- in second priority try EAP types in order of less leakage first; that is, EAP types with a configured outer identity should be tried before other EAP types which would reveal actual user identities.

6. Security Considerations

The security of an EAP conversation is determined by the EAP method which is used to authenticate. This document does not change the actual authentication with an EAP method, and all the security properties of the chosen EAP method remain. The format requirements (character encoding) and operational considerations (re-try EAP with a different EAP-Response/Identity) do not lead to new or different security properties.

7. IANA Considerations

There are no IANA actions in this document.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.
- [RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.
- [RFC4284] Adrangi, F., Lortz, V., Bari, F., and P. Eronen, "Identity Selection Hints for the Extensible Authentication Protocol (EAP)", RFC 4284, January 2006.
- [RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, August 2008.

[RFC5448] Arkko, J., Lehtovirta, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", RFC 5448, May 2009.

[RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

Author's Address

Stefan Winter
Fondation RESTENA
6, rue Richard Coudenhove-Kalergi
Luxembourg 1359
LUXEMBOURG

Phone: +352 424409 1
Fax: +352 422473
EMail: stefan.winter@restena.lu
URI: <http://www.restena.lu>.